

Siamese Networks for Surveillance and Security

JONATHAN BOYLE

PhD Thesis

Supervisors:

Prof. James Ferryman

Dr. Hong Wei

Department of Computer Science

School of Mathematical, Physical, and Computational Sciences

University of Reading

November 2022

Abstract

This thesis investigates the usage of Siamese networks across three surveillance and security tasks for land border security. Siamese networks (also known as a twin-pair network) are a layout of neural networks that contain a segment that contains duplicated architecture and configuration parameters for feature extraction of two inputs, combining the outputs into one vector for comparison in a final set of layers to produce a similarity score. The effectiveness of multiple architectures of Siamese networks crafted from multiple generations of Convolutional Neural Networks and Residual Neural Networks are examined for side-profile vehicle classification and Differential Morphing Attack Detection (D-MAD), and with a novel architecture for trajectory similarity analysis.

The challenging domain of automated vehicle classification from pole-mounted roadway cameras from side-profile views is evaluated. Three Siamese networks based on existing non-Siamese architectures are proposed and compared against five existing methods on a novel and published dataset. The evaluation undertaken shows that the residual based Siamese network is able to outperform other state of the art methods on datasets with a small number of classes.

An end-to-end Siamese trajectory network framework is proposed for the purpose of trajectory similarity analysis in surveillance tasks. A deep feature auto-encoding network is used as part of a discriminative Siamese architecture to perform trajectory similarity analysis. The effectiveness of this method is evaluated on four challenging public real-world datasets containing both vehicle and pedestrian targets, and compared with five existing methods. The proposed method outperforms the existing methods on three of the four datasets.

Face morphing attacks pose an increasingly severe threat to automatic face recognition systems in border control environments. Three Siamese architectures built up from multiple generations of non-Siamese Convolutional and Residual Neural Networks for D-MAD are proposed, showing the effectiveness of these networks against a pre-established Convolutional architecture for Single-image Morphing Attack Detection (S-MAD). The residual network based architecture outperforms representative convolutional architectures from the literature, with the Siamese D-MAD architecture able to outperform its S-MAD variant.

Acknowledgements

The following individuals have provided content that is directly used in this thesis:

- **Murray Evans** - For initial labelling of the Make/Model datasets on the Reading Side Profile Dataset in Chapter 2, and the initial concept and initial code for the automated extraction of images.
- **Clemens Seibold** - For the generation of two of the three morphing methodologies (field morphing and triangle morphing) in Chapter 4 as well as the S-MAD VGG-19 results.

This thesis has been partially funded by the following European Union Seventh Framework Programme and Horizon 2020 projects:

- EU FP7 Project - EFFISEC - Efficient Integrated Security Checkpoints - Grant Agreement No. 217991
- EU FP7 Project - P5 - Privacy Preserving Perimeter Protection Project - Grant Agreement No. 312784
- EU Horizon 2020 Project - D4FLY - Detecting Document frauD and iDentity on the fly - Grant Agreement No. 833704

The following individuals are those who supported the author in other ways:

- **Dawn, Nigel, and Samantha Boyle** - My loving and supportive Mother, Father and Sister.
- **Sujinnor Aldermann** - For the times we spent through lockdowns and beyond.
- **Nicholas Gurr** - For all the hardware technical support and personal support.

The following are those who gave all the support they could during their time in this world:

- **Cynthia Joyce Mary Walker** - Grandmother who passed February 2016.
- **Imelda Feodora Gomes** - Great Aunt who passed in October 2020.
- **Amanda Margaret Lankfer** - Aunt who passed in October 2020.

Contents

Abstract	ii
Acknowledgements	iii
Contents	iv
List of Tables	viii
List of Figures	x
List of Publications	xiv
1 Introduction	1
1.1 Concepts of Neural Networks	2
1.1.1 Neurons and Activation Functions	2
1.1.2 Networks of Layers and Deep Learning	4
1.1.3 Training Neural Networks	5
1.1.4 Siamese Neural Network	7
1.2 Challenges	8
1.2.1 Vehicle Side-Profile Sub-Type and Make/Model Classification	8
1.2.2 Trajectory Pattern Extraction and Classification	10
1.2.3 Differential Face Morphing Attack Detection	11
1.3 Research Questions	12
1.4 Scientific Contributions	13

1.5	Dissertation Overview	14
2	Vehicle Side-Profile Sub-Type and Make/Model Classification	15
2.1	Background	15
2.2	Problem Definition	17
2.2.1	Mathematical Definitions	17
2.2.2	Reading Side Profile Vehicle Dataset	17
2.3	Convolutional Siamese Architecture	21
2.3.1	Siamese Classifier Architecture	21
2.3.2	Auto-Encoder Training	23
2.4	Results and Discussions	25
2.4.1	Evaluation Metrics	25
2.4.2	Compared Methods	26
2.4.3	Auto-Encoder Results	28
2.4.4	Sub-Type Results	31
2.4.5	Make/Model Results	37
2.5	Conclusions	44
3	Trajectory Pattern Extraction and Classification	45
3.1	Background	46
3.2	Problem Definition	46
3.2.1	Mathematical Definition	46
3.2.2	Datasets	47

3.3	Auto-Encoder Architecture	49
3.3.1	Mathematical Model	49
3.3.2	Determining the Layered Architecture	51
3.4	Siamese Neural Network Architecture	52
3.5	Training Methodology	53
3.6	Results and Discussions	54
3.7	Conclusions	62
4	Differential Face Morphing Detection	63
4.1	Background	64
4.2	Problem Definition	65
4.2.1	Mathematical Definition	65
4.2.2	Datasets	65
4.2.3	Evaluation Metrics	66
4.3	Siamese Model Description	68
4.4	Methodology	70
4.5	Results and Discussions	70
4.6	Deployment	76
4.7	Conclusions	77
5	Conclusions and Future Work	79
5.1	Vehicle Sub-Type and Make/Model Classification	80
5.2	Trajectory Pattern Extraction and Classification	82

5.3	Differential Morphing Attack Detection	83
5.4	Overall Conclusions and Future Work	84

List of Tables

2.1	Breakdown of image counts for Sub-Type classifications for Reading Side Profile Vehicle Dataset	18
2.2	Summary of Make/Model classes per Set (minimum number of exemplars) in Reading Side Profile Vehicle Dataset	18
2.3	Breakdown of image counts for Colour classifications for Reading Side Profile Vehicle Dataset	20
2.4	Auto-encoder Batch Size Trials for AlexNet, VGG19, and ResNet50 with Full Scale Output on 8 GB and 16 GB GPUs with NDADAM [16] optimiser	24
2.5	Auto-encoder Batch Size Trials for AlexNet, VGG19, and ResNet50 at 0.75 Scale Output on 8 GB and 16 GB GPUs with NDADAM [16] optimiser	24
2.6	Auto-encoder Batch Size Trials for AlexNet, VGG19, and ResNet50 at 0.5 Scale Output on 8 GB and 16 GB GPUs with NDADAM [16] optimiser	25
2.7	Results of auto-encoder training for side-profile vehicles	29
2.8	MSE results of Siamese Convolutional Networks on Vehicle Sub-Type Dataset	31
2.9	Breakdown of Siamese Vehicle Sub-Type Results by Class on Testing Set	33
2.10	Min/Max/Median of class statistics for Siamese Vehicle Sub-Type Classifiers compared against HoG-RBF-SVM [21]	35
2.11	MSE results of ResNet50 on Vehicle Make/Model Datasets	37
2.12	Min/Max/Median of class statistics for ResNet50 Siamese vehicle make/model classifiers compared with random forest and support vector machine models	41
2.13	Top/Bottom 10 Make/Model Set 20 Classes for ResNet50	42
2.14	Set 300 Classes Scores for ResNet50	44

3.1	Summary of all trajectory datasets' video structures	47
3.2	Dataset Trajectory Information	48
3.3	Hyperparameters for Trajectory Auto-Encoder Layer Architecture	52
3.4	The Final Hidden Layer Sizes of Auto-Encoders for Trajectory Datasets	52
3.5	Evaluation of the Output of Trajectory Siamese Networks with Models Trained on Each Dataset	54
3.6	Overall Scores of Trained Trajectory Siamese Models for all Datasets	55
3.7	Evaluation results of methods in terms of P , R and $F1$ scores; the higher the score, the better. Top two methods and tied second scores are shown in bold. . .	62
4.1	Results of Morphing Attack Detectors. All scores provided are error rates - lower values across all scores are better. Bold values indicate the results of the morphing type that the model was trained on.	71
4.2	Noise and Image Modification Testing of Models via StirTrace [95] on the Triangle Morphing Dataset	75

List of Figures

1.1	Graphs of Common Neural Network Activation Functions	3
1.2	Siamese Architecture Conceptual Diagram	7
1.3	Average images of Sub-Type classes for Reading Side Profile Vehicle Dataset in alphabetical order. Left is visual spectrum imagery, right is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].	8
1.4	Visualisations of Trajectories on Two Datasets - One Showing Vehicles and the Other Showing Pedestrians	10
1.5	Examples of Field, Triangle and Average Face-Morphing Techniques	11
2.1	Average images of Sub-Type classes for Reading Side Profile Vehicle Dataset in alphabetical order. Left is visual spectrum imagery, right is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].	18
2.2	Average images of Make/Model classes of Set 20 for Reading Side Profile Vehicle Dataset in alphabetical order by class name. Top is visual spectrum imagery, bottom is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].	19
2.3	Average images of Colour classes for Reading Side Profile Vehicle Dataset in alphabetical order by class name. Left is visual spectrum imagery, right is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].	20
2.4	Diagram of the Process to Extract the Reading Side Profile Vehicle Dataset Imagery	21
2.5	Siamese Feature Architectures for Vehicle Sub-Type and Make/Model classification	23
2.6	Epoch Training MSEs for Vehicle Auto-Encoders	29

2.7	Visualised Outputs of Vehicle Auto-Encoders for City Car Exemplar	29
2.8	Visualised Outputs of Vehicle Auto-Encoders for Saloon Car Exemplar	30
2.9	Visualised Outputs of Vehicle Auto-Encoders for People Carrier Exemplar	30
2.10	Visualised Outputs of Vehicle Auto-Encoders for Van Exemplar	31
2.11	Epoch Training MSEs for Siamese Convolutional Networks on Vehicle Sub-Type Dataset	32
2.12	Epoch Classification Statistics (Precision, Recall, and F1-Score) for Siamese Convolutional Networks on Vehicle Sub-Type Dataset	32
2.13	Confusion Matrices for Siamese Vehicle Sub-Type Classifiers on Testing Set	34
2.14	Boxplot of class statistics for Siamese Vehicle Sub-Type Classifiers. '+' indicates outliers	34
2.15	Vehicle ResNet50 Sub-Type Mis-Classification Examples	36
2.16	Epoch Training MSEs for ResNet50 on Vehicle Make/Model Datasets	38
2.17	Epoch Classification Statistics (Precision, Recall, and F1-Score) for ResNet50 on Vehicle Make/Model Datasets	38
2.18	Confusion matrices for ResNet50 on testing sets of all make/model sets. Set 100 - set 20 contain no class names due to the larger number of classes	39
2.19	Vehicle ResNet50 Make/Model Set 20 Mis-classification Examples from Worst 5 Classes	43
3.1	Representative Images and Trajectories of the Datasets used for Evaluation. (a) and (b) are from [66], with (c) from [70] and (d) from [23].	48
3.2	Trajectories Before and After Normalisation from the Traffic Junction Dataset	48
3.3	Proposed feature representation uses a neural network-based approach that employs the output of the smallest hidden layer (L_2) of a trained auto-encoder to represent trajectory information. $\mathbf{V}_{\mathfrak{x}_j}$: vectorisation of the data of the j th trajectory (\mathfrak{x}_j); L_1 : first hidden layer; L_3 : output layer.	50

3.4	Architecture of the full Siamese Trajectory Network. All layers are fully connected layers.	53
3.5	Confusion Matrices of Siamese Model on Same Trained Dataset	56
3.6	Classified trajectories of the Siamese models on the same trained dataset. Each trajectory is shaded with the darker end being the start and the lighter end being the end. The Y-axis shows classified class, and the colour specifies ground truth.	57
3.7	Confusion Matrices of Specifically Trained Models Tested on Other Datasets	58
3.8	Classified Trajectories of Models Trained on All Datasets	59
3.9	Confusion Matrices of Models Trained on All Datasets	59
3.10	Classifications of the model trained on all datasets (b) and the Students003 model (c) for classes 1-3 of Students003. The top (a) shows a top-down 2D view.	60
3.11	Visualised Trajectories of Specifically Trained Siamese Models when Tested on Other Datasets	61
4.1	Examples of the output of field, triangle, and average morphing techniques on three individuals from the FERET dataset [39], [40].	67
4.2	Siamese Differential Morphing Attack Detector Network Architecture with VGG-19 as the Feature Network	68
4.3	Siamese Feature Architectures for Differential Morphing Attack Detection	69
4.4	Score distributions for the Field Morphing (FM) dataset. Green represents a genuine label and red represents a morph label. The blue line is threshold t determined from validation set.	73
4.5	Score distributions for the Triangle Morphing (TM) dataset. Green represents a genuine label and red represents a morph label. The blue line is threshold t determined from validation set.	73
4.6	Score distributions for the Average Morphing (AM) dataset. Green represents a genuine label and red represents a morph label. The blue line is threshold t determined from validation set.	74

4.7	D4FLY Enrolment Kiosk (Left) and Biometric Corridor (Right)	76
4.8	D-MAD Software Architecture for D4FLY	77

List of Publications

The research presented in this thesis has in part been published in the following articles.

- [21] J. Boyle and J. Ferryman, “Vehicle subtype, make and model classification from side profile video,” in *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2015, pp. 1–6. DOI: 10.1109/AVSS.2015.7301783.
- [30] J. Boyle, T. Nawaz, and J. Ferryman, “Deep trajectory representation-based clustering for motion pattern extraction in videos,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078509.
- [37] M. Evans, J. Boyle, and J. Ferryman, “Vehicle Classification using Evolutionary Forests,” in *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, vol. 2, Jan. 1, 2012, pp. 387–393.
- [38] J. Boyle, T. Nawaz, and J. Ferryman, “Using Deep Siamese networks for trajectory analysis to extract motion patterns in videos,” *Electronics Letters*, vol. 58, no. 9, pp. 356–359, 2022, ISSN: 1350-911X. DOI: 10.1049/e112.12460.

“Vehicle subtype, make and model classification from side profile video” was a finalist for best paper award at the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) conference in 2015.

Chapter 1

Introduction

The concept of a Siamese network (also known as a twin-pair network) was first proposed by Bromley [1]. The original premise was to use a Time Delay Neural Network (TDNN) to extract a set of features from two hand-written signatures and then combine them with a distancing calculation to determine their similarity. The key aspect of this feature extraction phase that separated this concept from other neural networks of the time is that the weights used for each neuron are shared across both inputs. This concept mimics the more traditional approach of a separate feature extractor algorithm (e.g. such as Histogram of Oriented Gradients (HoG) [2]) and passing into a separate classifier. While the original approach was for a TDNN, the concept of shared weights across multiple inputs can easily be extrapolated and applied to other types of neural networks.

There are many different scenarios that such a network is suitable for - one common category would be for the verification of identity such as facial recognition [3] or iris recognition [4]. These scenarios are very important to the running of country borders - whether land, sea or air. However, there are many other areas within border surveillance for which this technique is suitable to be applied.

This thesis looks into the following three use cases to determine the effectiveness and suitability of Siamese networks for land border security and surveillance tasks:

- Vehicle Side-Profile Sub-Type and Make/Model Classification - the identification of sub-type (e.g. hatchback or estate) and make/model (e.g. Honda Civic) of an instance of a vehicle
- Trajectory Pattern Extraction and Classification - the extraction and classification of vehicle and pedestrian trajectories when traversing through an environment
- Differential Face Morphing Attack Detection - the detection of a face morphing attack (when two or more faces are merged into one) by comparison to a captured image of the individual trying to pass a biometric check.

This chapter is separated into the following sections. Section 1.1 introduces the relevant background of neural networks with respect to Siamese networks. Section 1.2 describes the scenarios that have been chosen for evaluating Siamese networks. Section 1.3 poses the specific research questions that are addressed throughout this thesis. A list of the scientific contributions and respective publications resulting from the work described in this thesis can be found in Section 1.4 with an overview of the remainder of the thesis in Section 1.5.

1.1 Concepts of Neural Networks

This section contains a background of the models and methodologies of neural networks used throughout this thesis: section 1.1.1 describes the basic neuron used throughout all models; section 1.1.2 describes how the neurons are organised into layers; section 1.1.3 describes the training methodology of neural networks; and section 1.1.4 describes a Siamese network in more detail, as well as how to modify a pre-existing network into one.

1.1.1 Neurons and Activation Functions

The neural networks that are used today are heavily based on the model proposed by McCulloch and Pitts [5]. This work [5] provided a mathematical description of the understanding of how neurons in real brains performed, and is the basis for most modern architectures used in machine learning; however, did not provide any methodology on how to train the neurons. The general principle behind a neuron is the summation of a set of inputs multiplied by a set of weights with the addition of a bias value. The neuron then outputs a binary value depending on if an activation threshold was met - current architectures remove this aspect of a threshold in favour of the simple output as floating-point value. The equation used in today's algorithms is as follows:

$$x = \sum_n^N (w_n I_n) + b \quad (1.1)$$

where x is the output of the neuron, N is the number of inputs to the neuron, w is the weight, I is the input, and b is the bias.

In place of the activation threshold, an activation function is often used instead. The equation above with no further calculations is classed as a linear activation function. This is not usable beyond a single layer of neurons due to the property that a combination of multiple linear functions can be reduced to a single linear function, eliminating any gains of chaining neurons. It is also not capable to be used as part of gradient descent/back-propagation (described in Section 1.1.3) for training a model as the derivative of the linear function will always be a

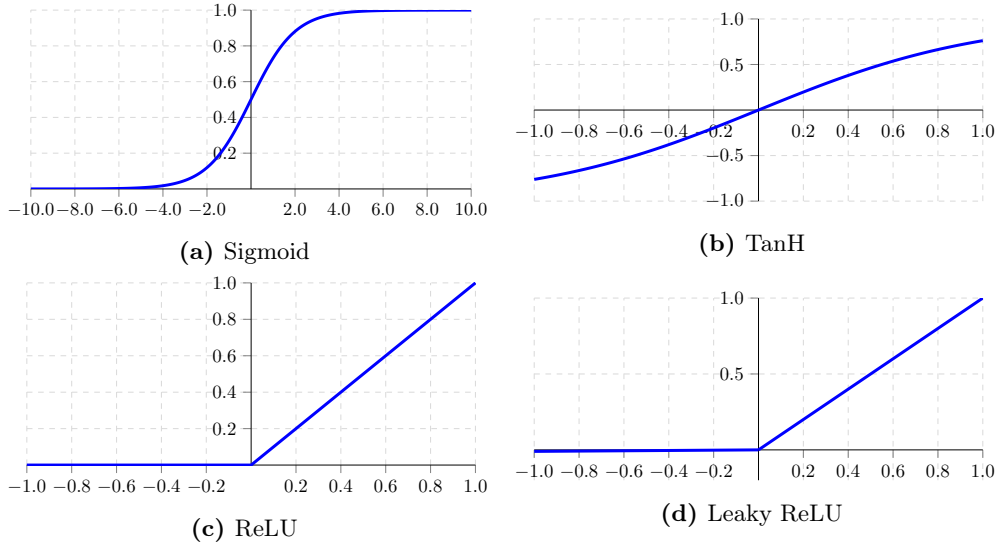


Figure 1.1: Graphs of Common Neural Network Activation Functions

constant. The linear activation would be defined as follows:

$$y = x \quad (1.2)$$

where x is the output from Eqtn. 1.1, and y is the overall output of the neuron (these will be the same for all activation functions). Typically, one of three activation functions are used in modern networks, each with their own benefits and negatives.

The first is a sigmoidal function, or a logistics curve (Figure 1.1a). The output of such a function is a smooth gradient in an ‘S’ shape between 0 and 1. One typical use for this type of function would be if the output is expected to be a probability. The mathematical equation is described below:

$$y = \frac{1}{1 + e^{-x}} \quad (1.3)$$

The second is a hyperbolic tan function - tanh (Figure 1.1b). This function looks overall very similar to the sigmoidal function, however has a different range: $[-1,1]$. As the output is centred around zero, the values can easily be mapped to strongly negative, neutral, or strongly positive. The mathematical equation is described below:

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (1.4)$$

Both sigmoidal and tanh activation functions suffer from the problem of a *vanishing gradient* - as the value of the input approaches either -3 or $+3$, the gradient approaches 0. If this happens, then any gradient descent algorithms used for training will no longer be able to optimise the neuron's weights. As such, it is best to attempt to ensure all values stay within a region of -1 and $+1$ by ensuring all inputs and outputs are within this range.

The third activation function is the Rectilinear Unit (ReLU) [6] which clamps the output value to a minimum of 0 (Figure 1.1c). While at a first look this might seem similar to the linear activation, it actually has a derivative making it possible to be used as part of back-propagation. Due to its simplicity, it is computationally very efficient and as such ensures a much faster learning time.

$$y = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1.5)$$

The default ReLU suffers from the *dying ReLU* problem. As all input values below 0 are ignored and no activation occurs, there is no gradient available to optimise for any of these values and as such cannot be trained if this occurs. This is partly solved by the Leaky ReLU that multiplies 0.01 to the input if below zero (Figure 1.1d) - although any of these values will take a long time to train:

$$y = \begin{cases} x, & \text{if } x > 0 \\ 0.01x, & \text{otherwise} \end{cases} \quad (1.6)$$

1.1.2 Networks of Layers and Deep Learning

Neurons are typically organised into layers - each layer consisting of multiple neurons all having the same input. All these neurons then produce outputs that can be organised into a vector that can either be used directly or output into another layer of neurons. If the neuron has a non-linear activation function this allows for more complex processing of the original input vector. A combination of layered neurons in such a way is called a 'neural network'.

Any layer that is not the final output layer is labelled as a 'hidden layer' for the purposes of learning/optimisation. The general principle behind training such a network is classed as supervised learning [7] - when one knows what outputs any given neuron should provide. In a

layered network however, any layer that is not the output layer would have an unknown output - such that it is 'hidden' in terms of the overall expectations of the optimisation problem.

Deep Learning as a term is used most frequently as a synonym of deep neural networks - a network with many layers. Schmidhuber [7] notes that even among experts there is no precise answer as to the difference between *shallow* learning and *deep* learning; however, defines that *very deep* learning has a *Credit Assignment Path* (CAP) of > 10 . CAPs are "chains of possibly causal links between events" in a network "from input through hidden to output layers" [7]. This can be viewed as approximately equal to the number of layers in a network, or the depth of calculations with modifiable variables. For the purpose of this thesis, Deep Learning will be defined as the training of a network with a CAP of > 2 .

In addition to the McCulloch Pitts neuron, there are plenty of other types of nodes used within modern deep networks. These typically are an entire layer of nodes of a different type, in the same feed-forward through layers manner. Convolutional layers revolutionised neural networks for image processing after 2012 on the ImageNet challenge [6], [8]. Instead of passing all the weights into a single neuron, a small bank of convolutional filters are used on patches across the input. Convolutional layers are often combined with different types of pooling layers in order to downscale the size of the output to decrease computational time.

While the most common way to connect layers is to have the output of one layer being piped into the next layer, there are other ways to connect them. Residual neural networks [9] pass the output of a layer not only to the next layer, but to a layer multiple steps ahead. This has been evidenced to be more performant than simply passing the layers onto the next, due to further layers having multiple sources of features to process. This also leads to less complex networks with smaller layers, which are also able to be far deeper and still have fewer parameters to optimise.

1.1.3 Training Neural Networks

The decades following the initial introduction of the architecture by McCulloch and Pitts [5] introduced the concepts of training neurons through both supervised and unsupervised methodologies [7], although these methods can be seen as variants of the linear regression methods introduced in the early 1800s. The most popular methodologies for training neural networks is based on Stochastic Gradient Descent (SGD) [10] and the back-propagation of errors [11], which is a multi-pass process that requires the processing of all the elements in a training dataset multiple times for a set amount of *epochs*.

The process of SGD attempts to optimise its problem by following the gradient of the activation function for a neuron to its minima - hence the requirement that an activation function must have some kind of mathematical derivative. The algorithm chooses a single sample for each iteration unlike traditional gradient descent which chooses computes for all samples. In the case of a neural network, one aims to optimise each of the weights for a neuron. First get the output of the neuron y , and calculate the overall error or loss ϵ relative to the correct expected output (e.g. the Mean-Squared-Errors).

$$\epsilon_n = (y_{o_n} - y_{p_n})^2 \quad (1.7)$$

$$\Delta w_n = \alpha * \frac{\delta \epsilon_n}{\delta y_{p_n}} \quad (1.8)$$

The derivative of this error is calculated based on the activation function and multiplied by a learning rate α to determine an update vector for the weights and biases of the neuron. A variant of SGD adds a momentum value γ multiplied by the previous $\delta \epsilon$ to speed up learning and to help stop the optimisation getting stuck in local minima.

$$\Delta w_n = (\alpha * \frac{\delta \epsilon_n}{\delta y_{p_n}}) + (\gamma * \Delta w_n^{(t-1)}) \quad (1.9)$$

When it comes to hidden layers (layers that are not the overall output, that cannot be seen), the correct expected output y_{o_n} is unknown and is therefore calculated by back-propagating the errors throughout the network. This works by the summation of all the changes to each weight from the entire set of inputs (usually split into batches) and propagating that error back to the previous layers' neurons.

Many extensions to SGD have been made over the years. The following is a list of some of the most notable:

- **ADAGRAD** [12] - Updates the learning rate as it progresses through the optimisation process depending on the difference between parameters over iterations/epochs. However, it suffers from an aggressive decrease in learning rate over time until it becomes extremely small and is no longer capable of learning.
- **ADADELTA** [13] - A more robust version of ADAGRAD attempting to deal with its

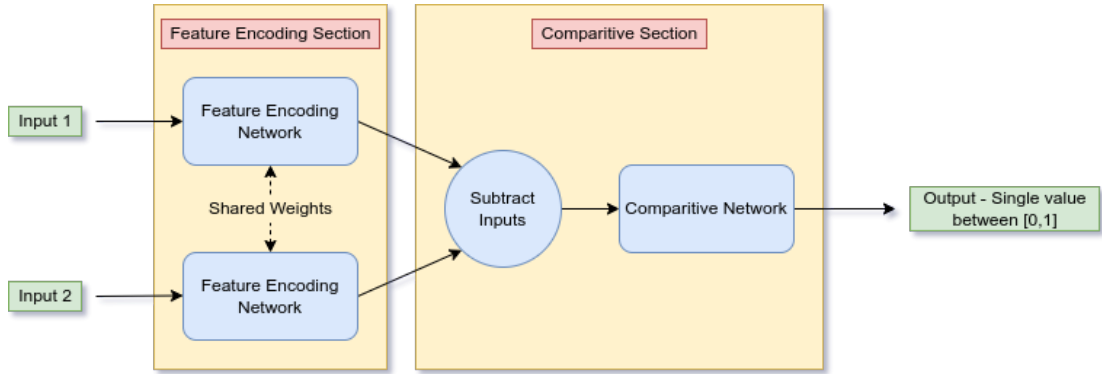


Figure 1.2: Siamese Architecture Conceptual Diagram

shortcomings by also calculating the updates with the difference of gradients.

- **ADAM** [14] - To the author’s knowledge, this is the most commonly used optimiser along with its derivatives. It takes advantage of the positives of all the optimisers above and more (such as RMSPROP [15]). While it is able to compute and learn significantly faster than SGD and above variants - however, the overall output tends to have less generalisation ability than standard SGD.

A number of variants of ADAM have appeared over the years. Normalized Direction Preserving ADAM (NDADAM) [16] attempts to fix the generalisation issues of ADAM by multiple means, by adapting the learning rate and other parameters to each weight vector rather than the entire network, and by a few other methodologies. NDADAM is the primary optimiser used throughout this thesis.

1.1.4 Siamese Neural Network

A Siamese network [1] comprises of two distinct sections: the feature encoding section and the final comparative section. Figure 1.2 shows a diagram depicting this process.

The feature encoding section is the first section of the network - it comprises of a series of layers that pre-processes multiple inputs. These layers are duplicated for each input - however, the weights and configuration parameters are identical in order to have the same feature encoding process. An alternative way to save on GPU memory at the cost of processing efficiency, if necessary, is to only have one set of layers and pass each input through it before passing both forward onto the comparative section.

Once the multiple inputs have been processed during the feature encoding sections, it is passed into the comparative section. This traditionally starts with a differencing or subtraction of the multiple vectors into a single one before being fed forward into the final layer(s). This ensures



Figure 1.3: Average images of Sub-Type classes for Reading Side Profile Vehicle Dataset in alphabetical order. Left is visual spectrum imagery, right is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].

that the overall network will process input from both inputs as all values have been merged. The final layer(s) then process the values into a final single value between $[0,1]$ determining similarity between the two inputs to the network.

Training the whole network with the inputs and expected output is a possible and viable method for some datasets. It is also possible to split the network into its separate sections and train them separately - allowing to control the encoding step more carefully. The encoding section can be trained as an auto-encoder [17], such as in Chapters 2 and 3 (a network that trains to reproduce its own inputs), to pre-train this functionality. The weights of this section can then be locked, and final training on the overall network (excluding the final layer nodes of the auto-encoder) can be performed.

Most neural networks and their variants can be converted into a Siamese network. The general process is to remove the final output layers defining the output size - for instance, in the case of a classifier it would have a number of output neurons the same as the number of classes. Once this layer has been removed, it can then be duplicated into a full feature encoding section with the comparison section placed at the end.

1.2 Challenges

The challenges for the three addressed scenarios are described in this section: section 1.2.1 describes the vehicle side-profile sub-type and make/model problem; section 1.2.2 describes the problems within trajectory pattern extraction and classification; and section 1.2.3 describes the overall challenges in face morphing attack detection.

1.2.1 Vehicle Side-Profile Sub-Type and Make/Model Classification

It is often necessary to identify a vehicle - whether passing through a land border checkpoint, entering a restricted facility, or on ordinary roads. The purpose of such a system varies with each application. A land border or facility entrance requires knowledge that the vehicle is permitted to

pass/enter. There are many tasks on ordinary roads: from identifying vehicles that are speeding or passing traffic lights illegally, to charging for car parks or congestion charges, as well as many more. Many of these systems involve enforcing fines on drivers that fail to follow the rules, so it is not surprising that people attempt to subvert the rules. Registered Traveller Programs (RTPs) can be used to pre-enrol travellers prior to arriving at the border in order to decrease the time to pass checks - automated vehicle sub-type and make/model classification can therefore be used to confirm the details of the travellers crossing the border.

The most widespread methodology to perform this task is by automated number-plate recognition (ANPR). Once the number-plate is recognised, it is looked up in a database to determine the details required for the specific task. However, this method is easily subvertable by the process of cloning number-plates. Reports suggest that this is an increasingly more common occurrence, with rising number of cases: it has been reported by the UK's Driver and Vehicle Licensing Agency (DVLA) that cases rose from 1,255 in 2012/13 to 4,802 in 2018/19 [19] - the AA (a UK vehicle insurance company) reports that the total number reported stolen in 2018 being 29,256 [20] with an estimated approximately 90,000 vehicles with cloned number plates on the road in that year.

While a number of cases are of the same type of vehicle, a large number are also reported to be of completely different types of vehicles: for example, number-plates for lorries being given speeding tickets with a picture of the vehicle provided being a sports car. As such, this is an important topic for research as it has wide-ranging effects on society.

The most common form of camera for capturing vehicles is a pole-mounted camera situated above the vehicle - however, the most discriminative viewpoint for identifying vehicle features is actually the side-profile perspective [21]. For border checkpoints and entrances to restricted facilities, it is often easier to install cameras from this perspective due to the lower height. However, to the best of the author's knowledge, this is an area not often explored in the academic literature despite its advantages.

The vehicle sub-type refers to the overall type of vehicle based on UK categories such as: hatchback, or saloon. These classes tend not to change, however the designs of vehicles still change over years: for instance, cars in the 2020s have far more curves in the design compared to ones from the 1980s. Due to the similarity-based component of the Siamese network, a classifier can be built similarly to a clustering algorithm: how close is an item to a centroid. This centroid (or average image) can be updated without the need to update the network - as such, could be adaptable to new vehicles with minimal to no training. Examples of average images for these

classes can be seen in Figure 1.3; further descriptions can be found in Chapter 2.

There are significantly more classes for the make/model of vehicles due to new models of vehicles coming out every year. This is challenging for a traditional classifier, as when a new model of vehicle is released it has to be re-trained to include the new models. A Siamese network is a similarity-based model that compares two inputs - as such, if it is able to train for one set of classes, it may need minimal or no training to extend as more vehicles get released to the public.

1.2.2 Trajectory Pattern Extraction and Classification

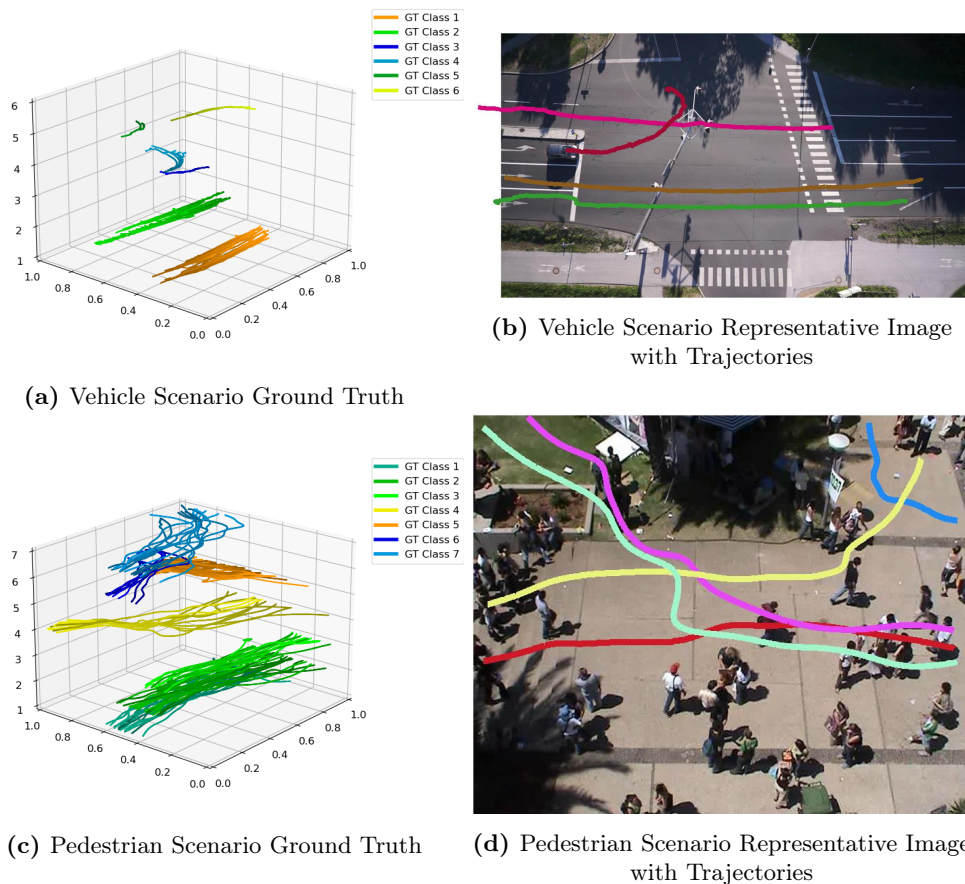


Figure 1.4: Visualisations of Trajectories on Two Datasets - One Showing Vehicles and the Other Showing Pedestrians

There are many possible tasks where trajectory analysis of some form is a useful tool in land border security and surveillance. A trajectory in this case is defined as a series of points often on a flat two-dimensional plane describing the movement pattern of some kind of tracked object target in a scene. The tracked object itself can be anything: from pedestrians, personal transportation vehicles on roads, boats, planes, or any other identifiable target. These extracted motion patterns can aid in activities such as: activity analysis [22], behaviour prediction [23], tracking [24] and abnormality detection [25], [26], trajectory prediction [27]–[29].

Different challenges arise depending on the type of target and scenario. Pedestrians walking through a train station have a very different type of trajectory to vehicles at a traffic junction. People are capable of changing directions rapidly and destinations can be reached from largely different paths at similar speeds. Vehicles tend to have smooth overall trajectories due to not being able to turn on the spot, but can accelerate to much faster velocities. Figure 1.4 shows example visualisations of the trajectories of a vehicle scenario and a pedestrians scenario from two of the four datasets used in Chapter 3.

There are two areas of analysis that can be performed on trajectories. Firstly the extraction of motion patterns from larger sets of trajectories to infer clusters of motion to identify common behaviours. Secondly, the subsequent classification task of assigning trajectories to these clusters is performed to both detect these common behaviours as well as anomalous behaviours.

It has been shown that a deep neural network model can be used to successfully encode these motion patterns into a feature vector that can successfully be clustered [30]. As a Siamese network contains a portion of the network that extracts features with the same weights, it is possible to convert such an encoding model to a Siamese network to take the place of a clustering algorithm for final classification.

1.2.3 Differential Face Morphing Attack Detection



Figure 1.5: Examples of Field, Triangle and Average Face-Morphing Techniques

Identifying individuals is a common task at borders. Traditionally, stations occupied by border control staff would manually check documentation from each traveller to identify if the

person in front of them is actually the individual described. This is an overall slow process with the potential for human error, and therefore more recently Automated Border Control (ABC) eGates have been deployed to verify individuals from biometrics stored in their electronic Machine Travel Document (eMRTD) such as an ePassport.

Technology has developed to the point that creating a face morph is now possible - where the merging of photos of two individuals can create an image that has the clear possibility of fooling both automated systems and manual checks [31] - allowing both to pass through the border with the same passport. It was classified as a vulnerability in ISO-19792 [32]. The first simple approach to perform this attack on biometric systems was published by Ferrara et al. [33] - with the first detector being proposed by Raghavendra [34]. Examples of some face morphing techniques can be found in Figure 1.5.

There are generally two categories of methodologies that face morph attack detection falls into: single image (when a method checks only the potential morph) and differential (when a potential morph is compared against a live individual). The land border control environment provides the opportunity to exploit not only the image in the passport but the traveller attempting to cross the border.

One of the primary cases for the use of Siamese networks has been in the role of face recognition [35], [36]. A Siamese network is trained up to be a discriminator between faces and furthermore no longer needs to be trained to identify individuals it has not seen. The problem of facial recognition is a similar problem to differential morphing attack detection in that they both take images of a live individual to compare against a reference image. The requirement of two images for comparison for the use of a Siamese network are therefore met for the problem of differential morphing attack detection.

As such, the land border scenario provides the requirements for both D-MAD and the use of Siamese networks.

1.3 Research Questions

The primary research question for this thesis is as follows:

- Are Siamese networks an appropriate and effective tool for the following security and surveillance tasks?
 - Vehicle Sub-Type and Make/Model Classification

- Trajectory Pattern Extraction and Classification
- Differential Morphing Attack Detection

In order to answer this question thoroughly, it needs to be broken down into parts for each particular scenario.

- Is a Siamese network an appropriate tool for the scenario? i.e. is it capable of performing robustly in the context of the defined task?
- Is a particular model able to outperform comparable/state-of-the-art methodologies as determined using appropriate benchmarking criteria?

1.4 Scientific Contributions

This section describes the overall scientific contributions contained within this thesis and relevant publications. The list is separated into the three main contribution chapters:

- Chapter 2 - Vehicle Side-Profile Sub-Type and Make/Model Classification
 - Generation of a novel Vehicle Side Profile Public Dataset - containing labels for approximately 10,000 images for: make/model, vehicle sub-type and colour.
 - Evaluation of multiple Siamese networks on dataset for make/model and sub-type based on pre-existing networks with comparison against multiple high-performing classifiers.
 - Two Publications:
 - * One primary author publication - “Vehicle subtype, make and model classification from side profile video” [21] - contributions for describing the automated extraction process for above dataset, alongside comparison of multiple classifiers.
 - * One secondary author publication - “Vehicle Classification using Evolutionary Forests” [37] - a contribution of analysis for novel evolutionary forest.
- Chapter 3 - Trajectory Pattern Extraction and Classification
 - Creation and testing of novel trajectory Siamese network with comparison to state-of-the-art methodologies.
 - Evaluation of trajectory Siamese networks on unseen datasets.
 - Two primary author publications:

- * “Deep trajectory representation-based clustering for motion pattern extraction in videos” [30] - a novel network design for neural network based auto-encoder using McCulloch Pitts neurons.
 - * “Using Deep Siamese networks for trajectory analysis to extract motion patterns in videos” [38] - the use of an auto-encoder network as the feature section of Siamese network.
- Chapter 4 - Differential Face Morphing Detection
 - Evaluation of multiple Siamese networks on multiple differential morphing attack techniques.
 - Morphing attack dataset generated from FERET [39], [40] of three pre-existing morphing techniques. Furthermore, the dataset has been designed to conform to ICAO specifications on machine readable travel documents.
 - Use within live environment for passenger enrolment in EU Horizon 2020 D4FLY project.

1.5 Dissertation Overview

This thesis is split into three main contribution chapters - each containing their own sections on literature, methodology, results, analysis and summary:

- Vehicle Side-Profile Sub-Type and Make/Model Classification (Chapter 2)
- Trajectory Pattern Extraction and Classification (Chapter 3)
- Differential Face Morphing Detection (Chapter 4)

Conclusions and future work can be found in Chapter 5.

Chapter 2

Vehicle Side-Profile Sub-Type and Make/Model Classification

Vehicle Sub-Type and Make/Model classification is a well studied problem in the literature as it has many important applications - such as traffic analysis, tolling and border control at border checkpoints. The top-down view of vehicles has issues with the frontal view being occluded of vehicles in flowing and queueing traffic [21] of which the side-view does not suffer from. While many situations a side-view of vehicles is not possible, at entrances to property and border checkpoints this is a far cheaper solution due to positioning at lower height.

A significant issue of any type of vehicle classification system is that there are consistently new models of vehicles being produced over time. Any system that is crafted to perform this task will require constant updating with newer models, and traditional machine learning techniques are often only trained to support a specific number of classes. Using a Siamese network is one approach to this problem - as the goal of a network is to detect similarity between inputs, it has the potential to either be used on newer models without or with very little extra training. This property of Siamese networks makes it a good option to explore - with three different architectures based on different generations of Convolutional Neural Networks examined in this chapter.

This chapter first provides an overview of related work in Section 2.1 with a definition of the problem in Section 2.2; the dataset that has been generated for the study of side-profile vehicle classification is described in Section 2.2.2; a description of the network architectures can be found in Section 2.3; and the results and analysis can be found in Section 2.4 with final conclusions in Section 2.5.

2.1 Background

The majority of research for computer vision based vehicle identification and classification is targeted towards the purpose of traffic analysis - and as such are based on a top-down viewpoint.

Prior work on video-based vehicle classification is generally limited to a small number of vehicle types and/or based on the American categories of vehicles that are typically larger which have a far higher inter-class variance.

Various approaches have studied the use of simple image analysis. Simple pixel quantity based on background subtraction has been used [41] to identify vehicle type on top-down imagery, with class types such as car/truck/bus which have a large difference in size. Other early studies tried to estimate the physical size of detected vehicles [42] or structural elements of said vehicles [43] to determine vehicle type out of three classes. [37] proposed an evolutionary random forest based on simple image features on side-profile imagery. Focusing in on a set of features is another approach that has been investigated. An approach using sixteen section direction chain code [44] identifies sub-type and model with a high accuracy by evaluating the front lamp contours across approximately 10,000 images, although has a limited analysis and provides little information on the dataset.

Studies have investigated different types of low level image features (such as SIFT and HoG). [45] applied various classifiers to three vehicle type classes based on SURF and Gabor features. [46] applied a two-step kNN classifier with geometric and texture based features for 7 classes of vehicle. [43] developed an SVM classifier based on a structural edge signature extracted from rear vehicle views, although limited to 3 classes and a small number of images (1664). [47] investigated a large number of features, however only had types of 2/3/4 wheeled vehicles and heavy vehicles. Different edge filters/detectors have been studied to determine if they can provide a better input for CNNs for determining type [48] - the only car sub-types being sedan and SUV while comparing against larger road vehicles. [49] modifies a ResNet50 architecture to outperform other residual and convolutional neural networks for classification on the CompCars dataset [50], which includes the hatchback alongside the sedan, MPV, and SUV.

Make/model recognition is a more challenging task than sub-type due to many car bodies looking physically similar. Some companies also share car bodies with their own badges, such as the Peugeot 107, Citroen C1 and the Toyota Aygo [51] and numerous others. Make/model recognition has overall far fewer studies to the author's knowledge. [52] applies number plate detection to captured frontal images of vehicles and trains a classifier ensemble based on the Gabor transform and PHoG for 600 images/15 brands/21 vehicle classes. [21] compares multiple SVMs based on HoG as well as multiple random forests across 86 classes for approximately 10,000 images on side-profile imagery. [53] used SIFT features as part of a bag-of-words model, and improved the bag-of-expressions approach using multiple keypoint detectors and HoG with a

SVM [54]. [55] uses a shallow CNN to classify 24 models of vehicles.

Excluding [21], [37], all the above methods focus on the top-down viewpoint of vehicles.

The topic of Siamese networks to the best of the author’s knowledge is not well explored within the literature of vehicle analysis. Twin Siamese networks have been used to look at different parts of the vehicle for the purpose of re-identification of a vehicle across multiple non-overlapping cameras [56].

2.2 Problem Definition

This section contains the mathematical definition of the problem in Section 2.2.1 as well as the dataset used for evaluation in Section 2.2.2

2.2.1 Mathematical Definitions

Let x_{v_i} be a side-profile image of the vehicle to be classified - this should be already normalised via the method described in Section 2.2.2, and x_{v_l} the manually defined label for the exemplar. x_{c_n} is an example or average of a class to determine a score for, where n is the ID of the class an N is the total number of classes $n \in \{1, \dots, N\}$ (a class can be either a vehicle sub-type or vehicle make/model depending on the scenario). The overall objective of the problem is to determine a predicted label Y the same as the input label x_{v_l} .

2.2.2 Reading Side Profile Vehicle Dataset

The Reading Side Profile Vehicle Dataset (RSPVD) contains imagery of over 10,000 instances of vehicles captured from a side profile [21] that was developed during the EFFISEC project [57]. The objective of the project was to provide border officials with up-to-date technologies for the purpose of being able to perform more in-depth checks as efficiently as possible.

A camera was installed to monitor the Pepper Lane entrance of the University of Reading with optical axis perpendicular to the flow of traffic at a height of approximately 2.5 m. At this height, the vehicle is seen primarily in profile, and the visibility of roof, bonnet (hood), and boot (trunk) surfaces is minimised. The camera was set to record two hours of the morning rush-hour every working day over the course of four weeks. Once vehicles had been detected and extracted, the resulting dataset contained over 10,000 images of vehicles, most of which were manually classified into sub-types as well as over 86 make/model categories with a wide range of populations. The dataset was later updated with colour labels for approximately a quarter of the dataset.

Sub-Type	Image Count
City	882
Estate	680
Hatchback	5821
Large Hatchback	694
People Carrier	215
Saloon	1074
Sports	189
SUV	357
Van	589
Total	10501

Table 2.1: Breakdown of image counts for Sub-Type classifications for Reading Side Profile Vehicle Dataset



Figure 2.1: Average images of Sub-Type classes for Reading Side Profile Vehicle Dataset in alphabetical order. Left is visual spectrum imagery, right is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].

A breakdown of the classes for sub-type can be found in Table 2.1 with average images for each class shown in Figure 2.1. The make/model categories were split into 4 separate sets due to a large disparity in the number of exemplars per class. Each set only contains vehicles with a make/model that has a minimum of N exemplars - Set 300, Set 100, Set 40, and Set 20; Table 2.2 contains a breakdown of these sets with Figure 2.2 showing averages of all classes. There are a total of 2541 images in the colour dataset, with a detailed breakdown in Table 2.3 with average images shown in Figure 2.3.

In order to detect a vehicle, wheel detection was performed under the presumption that from a side-profile a vehicle would have two visible wheels. A Random Forest classifier [58] was trained on 27674 wheel images manually extracted from the dataset and 13643 non-wheel images randomly extracted from the image (total 41319 images). Each wheel training image is a square image centred on the centre of the wheel, with a width to approximately match the diameter

Set	Number of Classes	Number of Images
300	6	3032
100	27	6460
40	59	8305
20	86	9141

Table 2.2: Summary of Make/Model classes per Set (minimum number of exemplars) in Reading Side Profile Vehicle Dataset

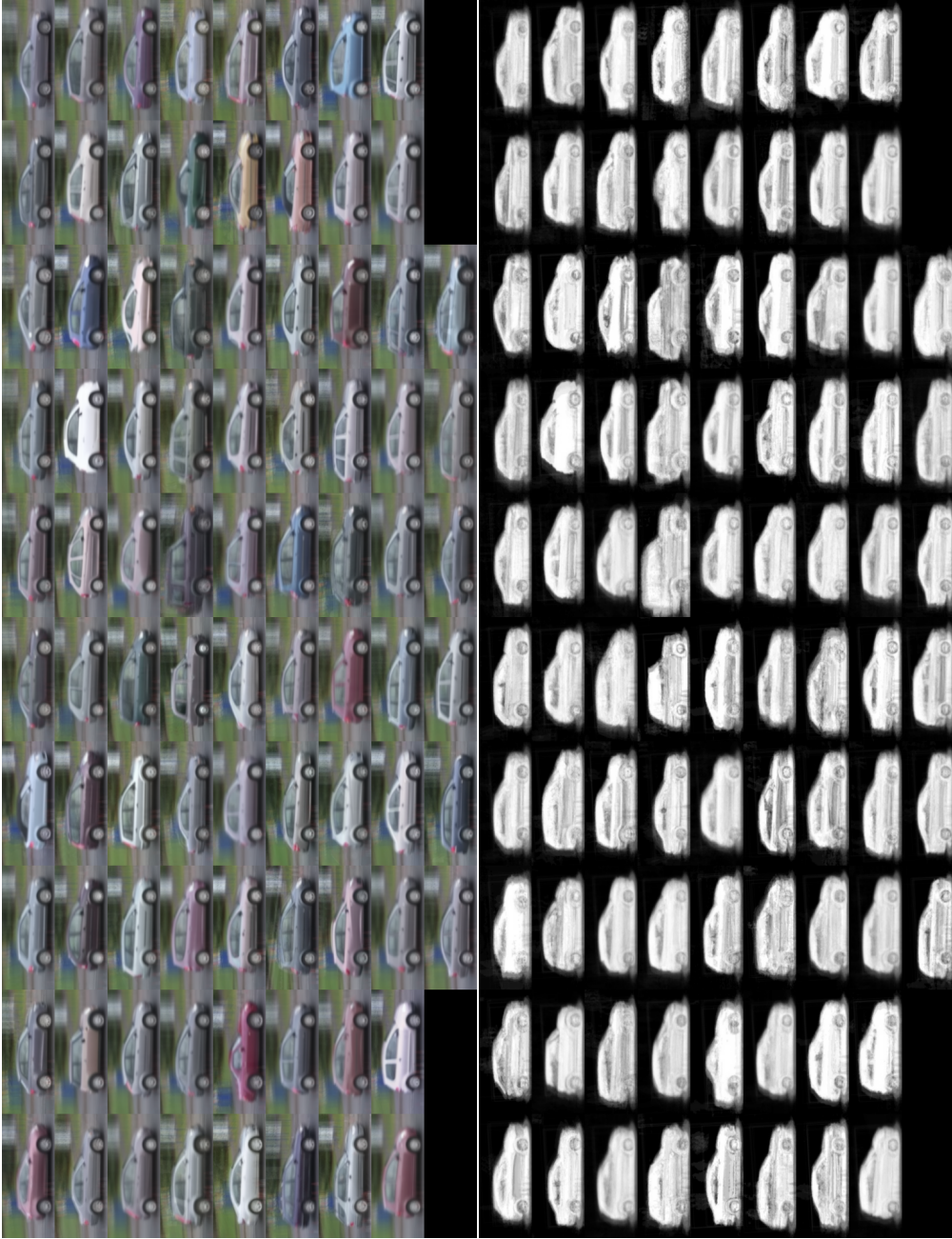


Figure 2.2: Average images of Make/Model classes of Set 20 for Reading Side Profile Vehicle Dataset in alphabetical order by class name. Top is visual spectrum imagery, bottom is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].

Colour	Image Count
Black	494
Blue	457
Green	167
Grey	194
Red	339
Silver	632
White	258
Total	2541

Table 2.3: Breakdown of image counts for Colour classifications for Reading Side Profile Vehicle Dataset



Figure 2.3: Average images of Colour classes for Reading Side Profile Vehicle Dataset in alphabetical order by class name. Left is visual spectrum imagery, right is generated binary background subtraction mask from using Adaptive Gaussian Mixture Models (AGMM) [18].

of the wheel plus some of the surround. The forest is grown in a traditional manner with three decision nodes - RGB colour difference, Gradient magnitude, Gradient direction. The final forest contained 798 trees.

A ‘trigger zone’ in the video is manually defined to detect when a vehicle first enters using an ‘Adaptive Gaussian Mixture Models (AGMM)’ change detector [18] to generate a binary background subtraction mask, from which the search for the front wheel occurs. Once discovered, a Stochastic Diffusion Search (SDS) is performed across five dimensions: x , y , d_x , d_y , s , where (x, y) describes the location of the front wheel, $(x + d_x, y + d_y)$ the location of the rear wheel, and s controls the size of the bounding box extracted from the image and used by the wheel classifier (the variation in wheel size between vehicle types can be large enough to warrant the search in scale). Letting v_w represent the votes of the classifier for the wheel class - and $v_{\bar{w}}$ represent the votes for non-wheel classes - the search seeks to maximize $v_w - v_{\bar{w}}$ for both wheels.

Each instance of a vehicle in the image is then rescaled and rotated to a resolution of 200x85 with the back wheel and front wheel positioned at 50 pixels from the edge of the image, with a motion mask and the scale factor used to resize the image. A diagram of the full process can be found in Figure 2.4.

This entire process was automated for the EFFISEC project [57] for the Albata land border between Romania and Moldova. The information was gathered on vehicles approaching the 1st

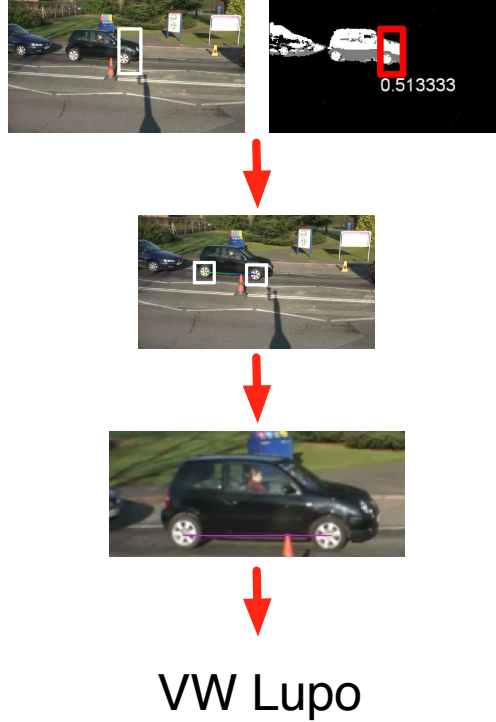


Figure 2.4: Diagram of the Process to Extract the Reading Side Profile Vehicle Dataset Imagery

security check to be forwarded onto the border guards to detect any anomalies. ANPR was performed to access the vehicle information from a database, for which it would be compared to the in-situ vehicle - any discrepancies were highlighted to the border guard on a custom terminal ready for the vehicle prior to it stopping at the check.

2.3 Convolutional Siamese Architecture

This section describes the overall architecture and the training methodologies. The Siamese architectures along with training can be found in Section 2.3.1. Section 2.3.2 describes the process of training an auto-encoder to assist with initial training.

2.3.1 Siamese Classifier Architecture

The concept of a Siamese network [1] is to create a network of two parts. The first part is a feature network f that is used to extract out feature vectors $F_{x_{v_i}}$ and $F_{x_{c_n}}$ from inputs x_{v_i} and x_{c_n} - f is used with the same architecture and weights for each input. The difference of these feature vectors ($F_{x_{v_i}} - F_{x_{c_n}}$) is then calculated to be passed towards a final layer O (or layers). In this case: the output layer O provides a single output y that predicts whether the test belongs to class n or not (Eqtn. 2.1).

$$y(x_{v_i}, x_{c_n}) = \begin{cases} 1 & \text{if } n = x_{v_i} \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

Once x_{v_i} has been tested for all n classes, a final label Y can be prescribed based on the maximum value of all y outputs for all values of n (Eqtn. 2.2).

$$Y(x_{v_i}) = \arg \min_{x_{v_i}} f(x_{v_i}) = \{x_{v_i} | \forall n : y(x_{v_i}, x_{c_n})\} \quad (2.2)$$

The process to convert an existing network into a ‘feature’ network f for a Siamese network involves modifying the original architecture to extract out the relevant parts. This differs for each existing network, however, a general rule of thumb is to either: remove the last layer and replace it with the remainder of the Siamese architecture, or use the network in its entirety. In the cases below, only the layer defining the number of outputs of the network has been replaced.

Three models representing different generations of CNNs have been selected as the base of a Siamese network for testing: AlexNet (Figure 2.5a), VGG-19 (Figure 2.5b), and the residual network ResNet50 (Figure 2.5c). The weights of each model have been pre-trained on ImageNet, and as such each image is normalised based on the ImageNet mean average per colour channel of 0.485, 0.456, 0.406 - and standard deviation of 0.229, 0.224, 0.225. Each image is also resized to 224x224 to match the input size of the each network. The models were then trained via the methods in Section 2.3.2, before being trained as a Siamese network for 100 epochs across the entire training set using the NDADAM optimiser with default learning rate of $1e^{-3}$.

The images in the dataset have been masked via the motion mask in the dataset. For each training epoch, a random selection of exemplars per class is chosen based on the minimum number of exemplars in the dataset in order to keep a balanced set of training among all the classes due to the large disparity in numbers between them. If one class has over 300 samples, and another has only 20 samples, the optimisation algorithm will likely optimise for the classes with larger samples.

In order to balance positive (similar) exemplars with negative (dissimilar) exemplars, only one negative exemplar is chosen. This is an attempt to improve recall - the ability of a model to correctly assign a sample to its class. With larger datasets such as Set 20, if one trains every possible combination, one will end up with 85 negative exemplars and 1 positive exemplar - an optimisation algorithm will likely learn this if it provides a ‘dissimilar’ score for every class.



Figure 2.5: Siamese Feature Architectures for Vehicle Sub-Type and Make/Model classification

In a non-Siamese configuration: the entire output for all classes are considered as one sample for the optimisation. In the Siamese configuration however 86 samples would be used - one for every comparison to a class average. The simple path for any optimisation algorithm would be to provide dissimilar scores for every output and it would have a MSE of $\frac{1}{86}^2 = 0.000135$ - a very low score with both recall/precision being close to or exactly 0.0. This effect is far less for datasets with a small amount of classes, however could still make a significant difference.

2.3.2 Auto-Encoder Training

An auto-encoder (also known as autoassociator or diabolo network) [17] is a type of network that is capable of reproducing its own inputs. The goal of the process is to have successively smaller layers to produce an efficient encoding of the input imagery before recreating the input image. In this case, the method is used as a pre-training of the network in order to train the network to understand the type of imagery that is input before doing any further training for the actual

NVIDIA GPU	Model	Maximum Possible Batch Size
2080 Super (8 GB)	AlexNet	Not Possible
	VGG19	Not Possible
	ResNet50	2
5000 (16 GB)	AlexNet	Not Possible
	VGG19	Not Possible
	ResNet50	32 (if no desktop loaded)

Table 2.4: Auto-encoder Batch Size Trials for AlexNet, VGG19, and ResNet50 with Full Scale Output on 8 GB and 16 GB GPUs with NDADAM [16] optimiser

NVIDIA GPU	Model	Maximum Possible Batch Size
2080 Super (8GB)	AlexNet	Not Possible
	VGG19	Not Possible
	ResNet50	16
5000 (16GB)	AlexNet	1024
	VGG19	32
	ResNet50	32

Table 2.5: Auto-encoder Batch Size Trials for AlexNet, VGG19, and ResNet50 at 0.75 Scale Output on 8 GB and 16 GB GPUs with NDADAM [16] optimiser

desired output. This method was chosen to pre-train the networks on the vehicle imagery.

The feature network f that is used for the Siamese architecture is temporarily modified to have a separate output layer O_a comprising of linearly activated neurons to reproduce the original input imagery - however, hardware limitations cause issues with this when it comes to this process. All the models tested had their input imagery at a resolution of 224x224 with 3 colour channels, requiring a total of 150,528 output nodes to recreate the image. Two GPUs were tested: NVIDIA GeForce RTX 2080 Super (8 GB RAM) and NVIDIA Quadro RTX 5000 (16 GB RAM). Both GPUs are incapable of training these networks (excluding ResNet50, although at a low batch size) using NDADAM [16] or other ADA based optimiser. Table 2.4 shows the possible batch sizes for each model across these two GPUs, with **Not Possible** indicating when the process exceeds a GPU’s RAM. In order to get the models to train, an experiment was performed using auto-encoders at different scales (i.e. reproduce the output at a smaller size) across 2 epochs of training to determine what would be possible. Table 2.5 shows the test on scaled output imagery by 0.75, and Table 2.6 shows the test on scaled output imagery by 0.5.

In order to keep a consistent training across all models, the output scale of 0.75 was chosen due to being able to run at 32 sized batches on the 16 GB GPU. Each model was trained for 200 epochs across the entire training set using the NDADAM optimiser with default learning rate of $1e^{-3}$.

NVIDIA GPU	Model	Maximum Possible Batch Size
2080 Super (8 GB)	AlexNet	256
	VGG19	8
	ResNet50	8
5000 (16 GB)	AlexNet	2048
	VGG19	32
	ResNet50	64

Table 2.6: Auto-encoder Batch Size Trials for AlexNet, VGG19, and ResNet50 at 0.5 Scale Output on 8 GB and 16 GB GPUs with NDADAM [16] optimiser

2.4 Results and Discussions

This section describes the results of the Siamese methodology described above, as well as comparing them against a number of other classifiers. The evaluation metrics can be found in Section 2.4.1, followed by a description of the methods used for comparison in Section 2.4.2.

The results for the auto-encoder training can be found in Section 2.4.3, followed by the results on the sub-type dataset in Section 2.4.4, and finishing with the make/model results in Section 2.4.5.

2.4.1 Evaluation Metrics

The metrics used to evaluate the results of all the training can be found below. Equation 2.3 shows the loss function used for training - the Mean Squared Error (MSE):

$$\text{MSE} = \sum_{i=1}^D \frac{(\mathfrak{x}_i - \mathcal{Y}_i)^2}{D} \quad (2.3)$$

where D is the number of samples, \mathfrak{x}_i is the output of the network on the i -th sample, and \mathcal{Y}_i is the true value of what the output of the network should be.

For the Sub-Type and Make/Model training: Precision (Eqtn. 2.4), Recall (Eqtn. 2.5) and F1-Score (Eqtn. 2.6) are used.

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.4)$$

$$R = \frac{\text{TP}}{P} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.5)$$

$$F_1 = 2 \times \frac{P \times R}{P + R} = \frac{2TP}{2TP + FP + FN} \quad (2.6)$$

In the three equations, TP stands for True Positives, FP stands for False Positives and FN stands for False Negatives. For multi-class evaluation, the weighted average is used to calculate the Precision/Recall/F1Score - the weights being calculated based on the support of the class (how many instances of the class in the testing set). As such, the weighted average statistics may not have an F1-Score in-between the Precision and Recall as described by the equations above.

2.4.2 Compared Methods

A total of five other types of models are used for the comparison:

1. **Img-Forest** - A random forest operating on simple image features [21], [37].

The concept of the random forest was first introduced by Breiman [58] in 2001: an ensemble of binary decision trees where each decision node of the tree is configured to exploit a randomly chosen classifier from a set of available classifiers. In this scenario, the dataset consists of RGB images and black/white foreground masks. The only pre-processing before being input into the learning algorithm (or after for classification) is to produce gradient magnitude and orientation images. The selection of potential nodes for the use of the random forest was first derived by Evans [37] and refined in [21]. These nodes are as follows:

- (a) **RGB colour difference** - Given two pre-specified pixel locations, threshold the L2 colour difference. This requires a total of 5 parameters: (x_1, y_1) , (x_2, y_2) and τ (the threshold).
- (b) **Gradient magnitude** - Given a single pixel coordinate (x, y) , determine if the gradient magnitude is larger than a threshold τ .
- (c) **Gradient orientation** - Given a single pixel coordinate (x, y) , determine if the gradient orientation is between two thresholds τ_1 and τ_2 .
- (d) **Mask pixels** - Given a single pixel coordinate (x, y) , determine if it is foreground or background.
- (e) **Scale Range** - Determine if the scale factor (derived as part of the dataset as described in Section 2.2.2) used when resizing the image is between two thresholds τ_1 and τ_2 .

The forest has a total of 200 grown trees, with a maximum depth of 20. A node is selected at random from the list above and optimised using Gini Impurity. Each tree is grown independently of each other.

2. **Evo-Forest** - An evolutionary forest that uses genetic algorithms to evolve a forest optimised for the task [21], [37].

The image forest optimises each tree to be the best a particular tree can be, however the traditional training methodology does not account for producing the best combination of trees in the total forest. An evolutionary approach has been used for the same type of trees as described above with the same types of decision nodes. The fitness function for the evolutionary approach attempts to ensure that each new tree maximises its own strength like above, while also minimising correlation with existing trees. The same number of 200 trees is imposed on the algorithm, with a termination condition to stop adding additional trees if the accuracy reaches 100%. The process of training an evolutionary forest consisting of n trees is summarised as follows:

- Start with an empty set F_f of trees.
- Iterate until $|F_f| = n$:
 - Create a sub-sample of the training set for this iteration.
 - Grow a pool of potential trees F_p .
 - Iterate until ready:
 - * Evaluate trees in pool.
 - * Replace poor trees with new trees by cross-breeding/mutating/growing.
 - Take the best tree in F_p and add to F_f .

3. **HoG-Forest** - A traditional random forest using a Histogram of Oriented Gradients (HoG) descriptor of the whole image.

The HoG descriptor consists of 7524 elements generated with a block size of (20, 14) pixels and a stride of (10, 7) from a greyscale version of the image. Only a single type of node is used with a threshold τ , and trained in the same way as the Img-Forest.

4. **HoG-Lin-SVM** - A linear Support Vector Machine (SVM) operating on the image's HoG descriptor.

Uses a one-vs-all multi-class SVM [59] with a linear kernel on the same HoG descriptor as described above. The SVM's C parameter was optimised in the range {10,100,1000}.

5. **HoG-RBF-SVM** - A Support Vector Machine (SVM) with an RBF kernel operating on the image's HoG descriptor.

The same type of SVM as above, however using an RBF kernel. Uses the same C parameter range above, as well as a search on γ in the range $\{0.0001, 0.001, 0.01\}$.

2.4.3 Auto-Encoder Results

The Mean Squared Errors (MSEs) of the auto-encoder training can be found in Table 2.7 with graphs of the values at each epoch in Figure 2.6. The ResNet50 model has clearly trained the best with much lower MSEs (test 0.161) compared to VGG-19 (test 0.242) and AlexNet (test 0.715), which is an overall expected result considering the age/generations of each model. Visualisations of two classes from the results can be found for a random exemplar from city car class (Figure 2.7), saloon car class (Figure 2.8), people carrier class (Figure 2.9) and van class (Figure 2.10).

It is clear from the MSE and the visualised output that AlexNet was unable to train further than a relatively similar image for all inputs and struggled to optimise its score throughout the entire training process. VGG-19 was able to understand the overall shapes, however had issues with recalling the overall outline and shape, but lacked a lot of colour compared to the pre-processed image - although is capable of bringing through some details with the van exemplar. In the case of the van exemplar - due to the dataset being recorded across multiple days and the van being marked as a University of Reading van (where the dataset was recorded) - that exemplars of the same vehicle appear in both training and testing sets. The ResNet50 model was able to recall the colours far better than the other models, and is capable of reproducing the overall shape better.

Issues arise for both the VGG-19 and ResNet50 models with darker-coloured vehicles such as Figure 2.9. The normalisation process based on the ImageNet mean/standard deviation causes a lot of detail of the vehicle to be lost due to being closer to the black background used for masking out the background. This can also be seen in the other visualisations - however it appears the worst for this particular vehicle.

Overall, it is clear that the ResNet50 model is the best of the three followed by the VGG-19 and AlexNet.

Model Type	Time to Train	Train Set MSE	Test Set MSE
AlexNet	1h 1m 23s	0.866	0.715
VGG-19	4h 57m 53s	0.126	0.242
ResNet50	2h 10m 4s	0.0765	0.161

Table 2.7: Results of auto-encoder training for side-profile vehicles

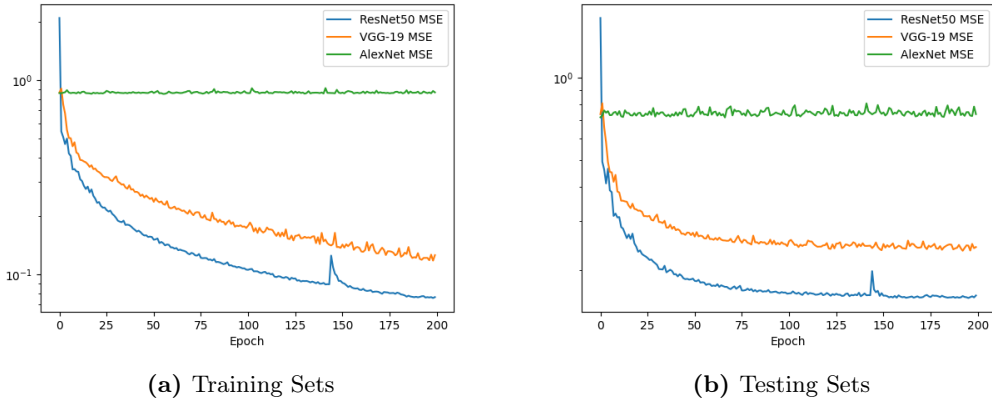


Figure 2.6: Epoch Training MSEs for Vehicle Auto-Encoders

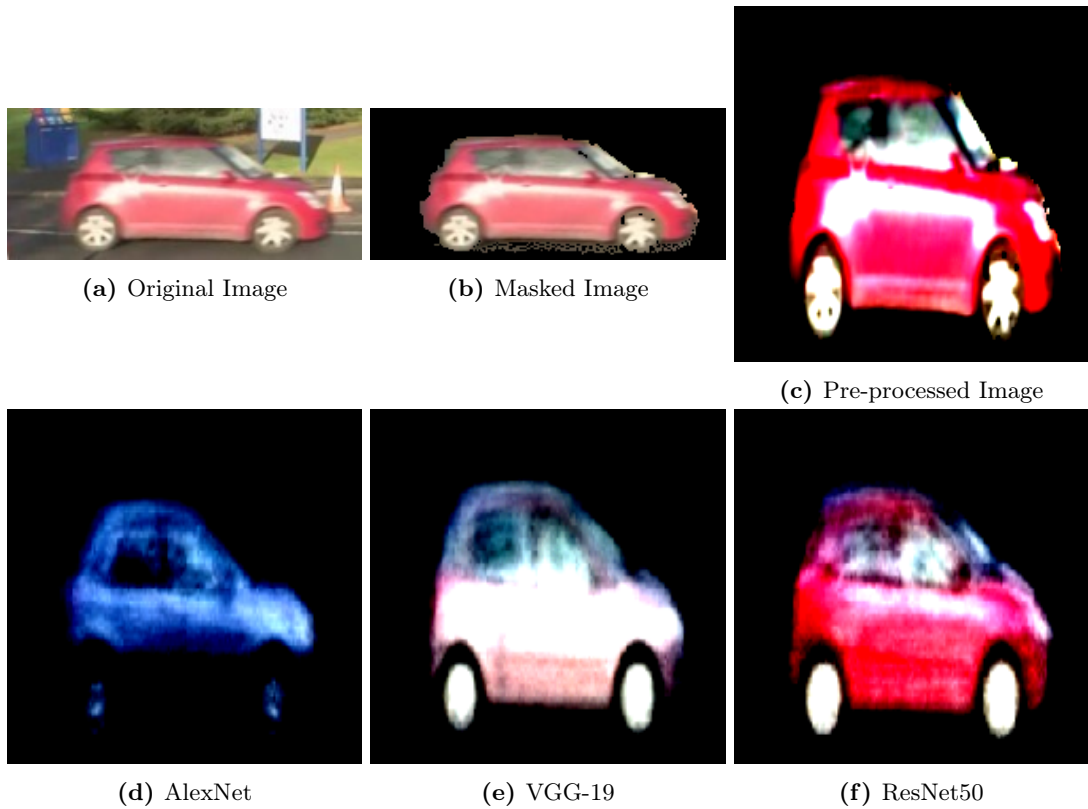


Figure 2.7: Visualised Outputs of Vehicle Auto-Encoders for City Car Exemplar

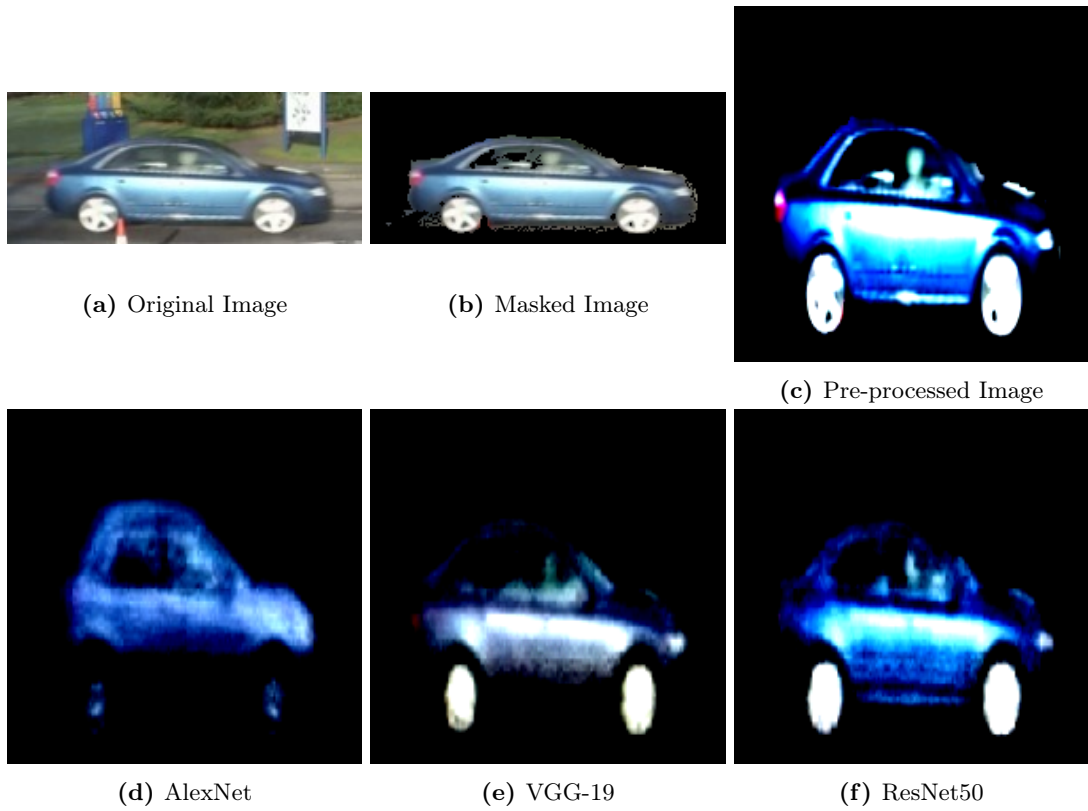


Figure 2.8: Visualised Outputs of Vehicle Auto-Encoders for Saloon Car Exemplar

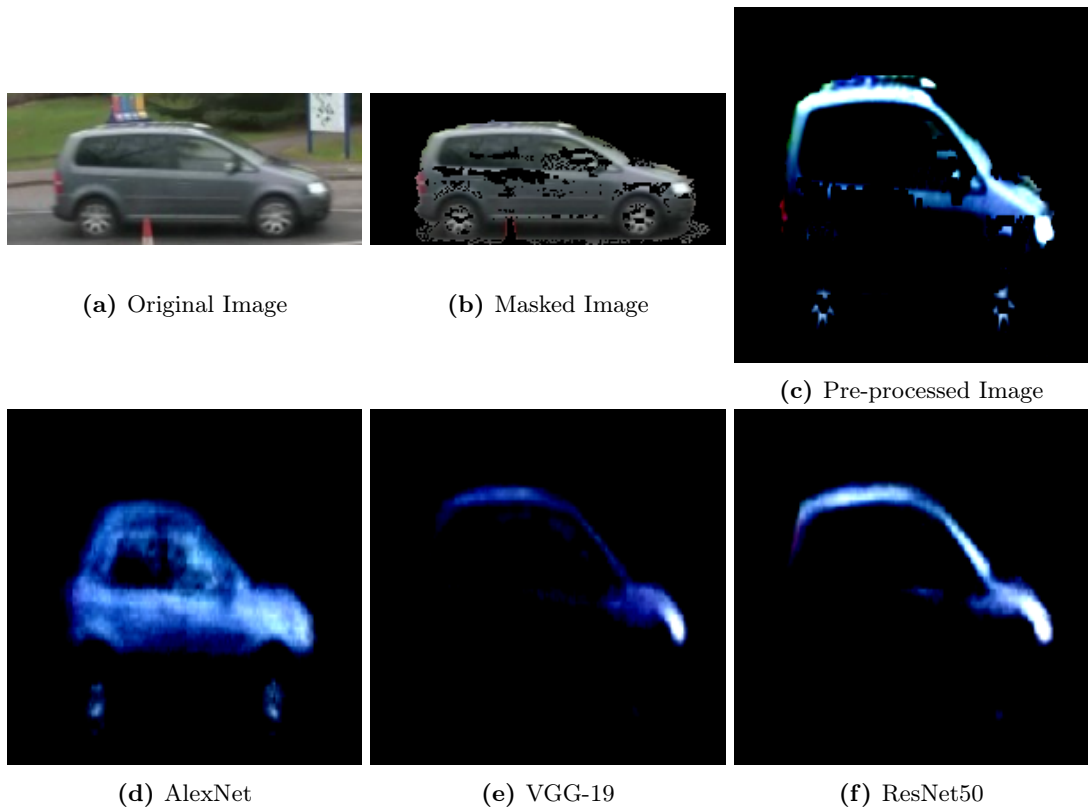


Figure 2.9: Visualised Outputs of Vehicle Auto-Encoders for People Carrier Exemplar

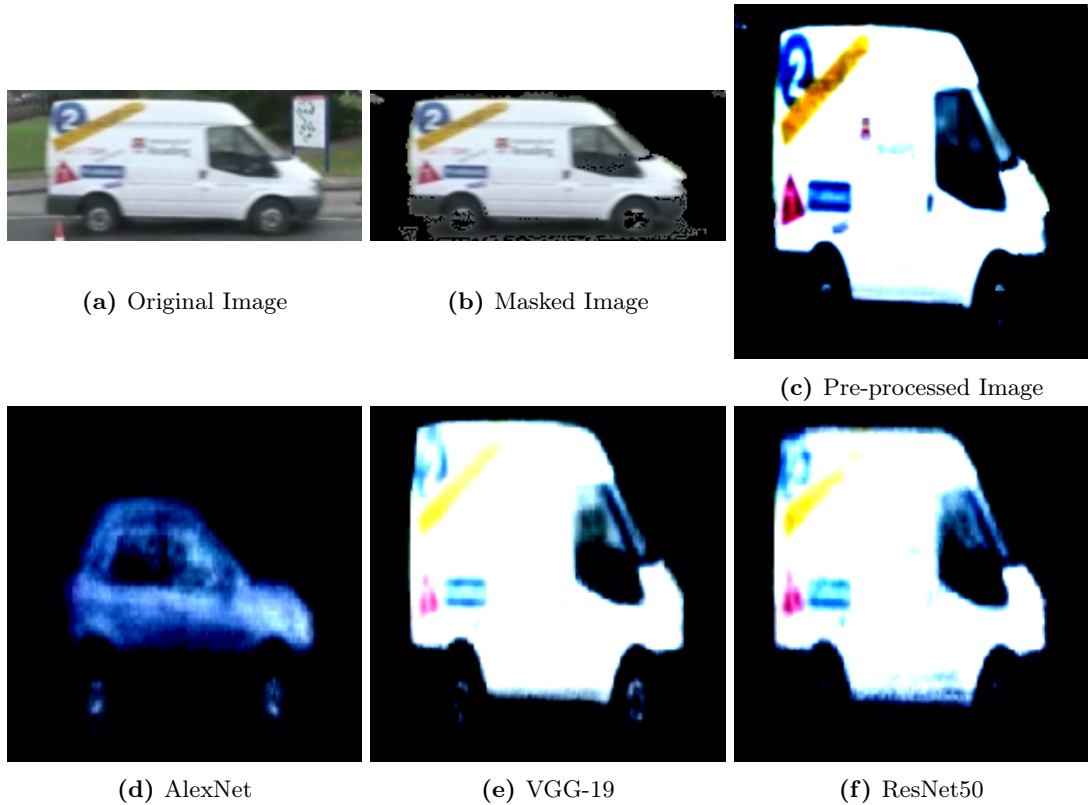


Figure 2.10: Visualised Outputs of Vehicle Auto-Encoders for Van Exemplar

2.4.4 Sub-Type Results

The MSEs of the training from the sub-type dataset can be found in Table 2.8. The testing set for ResNet50 was evaluated every 5 epochs, with the VGG-19 every 10 epochs and AlexNet every 20 epochs. Both the AlexNet and VGG-19 had issues training and hit the 0.988 mark without being able to further decrease in loss, whereas the ResNet50 was able to get substantially lower scores for the training set (0.000546) and testing set (0.0157).

Graphs depicting the MSEs per epoch can be found in Figure 2.11. A large amount of oscillating can be seen in the MSE graph for ResNet50 - this can in part be attributed to the use of randomly selecting exemplars based on the minimum class size as each epoch a different set of exemplars could pass through the training algorithm. The VGG-19 and AlexNet MSEs are however a lot more stable, although both have fallen into a minima in the optimisation gradients early on in the training. Eyeballing the various classification statistics in Figure 2.12 (precision,

Model Type	Time to Train	Train Set MSE	Test Set MSE
AlexNet	4h 12m 18s	0.0988	0.0988
VGG-19	2d 2h 50m 51s	0.0988	0.0988
ResNet50	1d 1h 34m 2s	0.000546	0.0157

Table 2.8: MSE results of Siamese Convolutional Networks on Vehicle Sub-Type Dataset

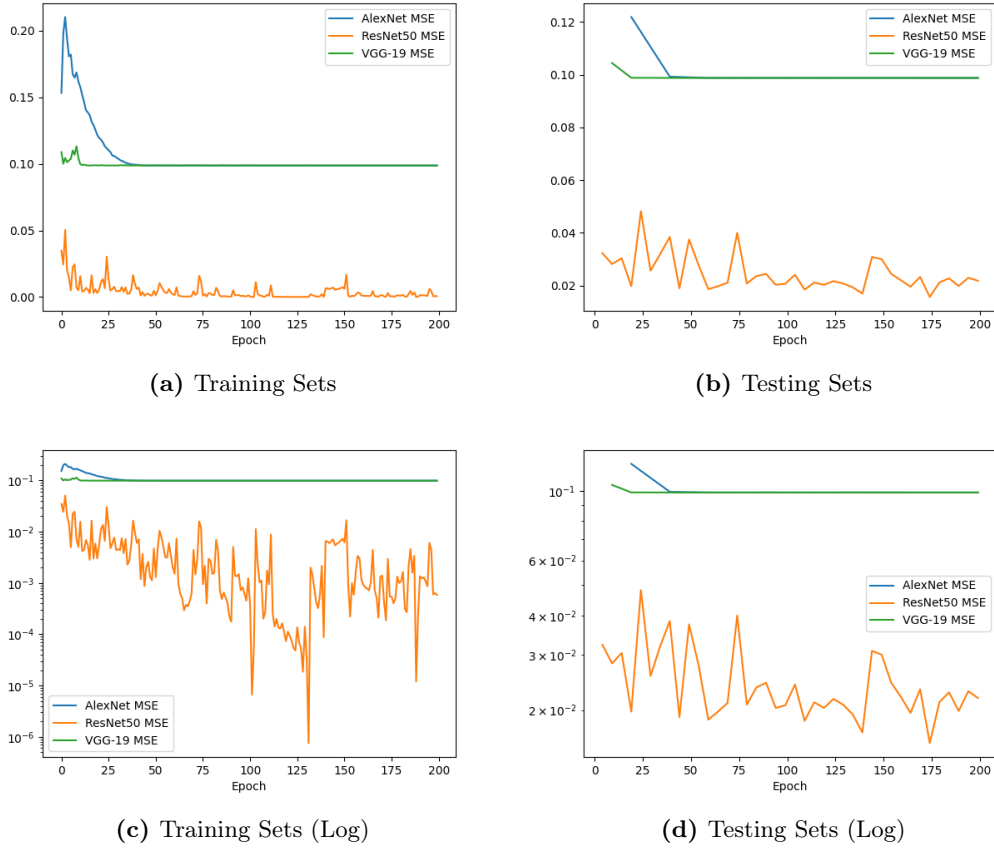


Figure 2.11: Epoch Training MSEs for Siamese Convolutional Networks on Vehicle Sub-Type Dataset

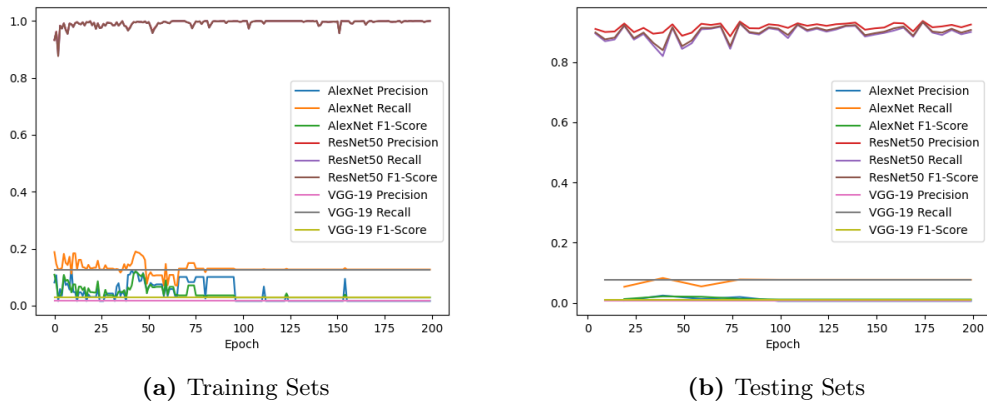


Figure 2.12: Epoch Classification Statistics (Precision, Recall, and F1-Score) for Siamese Convolutional Networks on Vehicle Sub-Type Dataset

Class	Model								
	AlexNet			VGG-19			ResNet50		
	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>
City	0.0765	0.997	0.142	0.0764	1	0.142	0.796	0.893	0.842
Estate	0	0	0	0	0	0	0.893	0.977	0.933
Hatchback	0	0	0	0	0	0	0.979	0.934	0.956
Largehatch	0	0	0	0	0	0	0.864	0.759	0.808
Peoplecarrier	0	0	0	0	0	0	0.69	0.926	0.791
Saloon	0	0	0	0	0	0	0.904	0.977	0.939
Sports	0	0	0	0	0	0	0.886	0.979	0.93
Suv	0	0	0	0	0	0	0.88	0.983	0.929
Van	0.306	0.0283	0.0518	0	0	0	0.886	0.997	0.938
Macro avg	0.0425	0.114	0.0215	0.00849	0.111	0.0158	0.864	0.936	0.896
Weighted avg	0.0192	0.0774	0.0131	0.00584	0.0764	0.0109	0.936	0.932	0.933

Table 2.9: Breakdown of Siamese Vehicle Sub-Type Results by Class on Testing Set

recall, and f1-score), it is clear that for the ResNet50 model, the training set was optimised to near perfect classification scores within the first few epochs. This was not replicated within the testing set, with scores sitting around the 0.9 boundary for the entire training cycle - although stabilising around the 100th epoch.

For the task of sub-type classification, only ResNet50 was able to train to perform well in classification. A breakdown of the classification statistics for the testing set can be found in Table 2.9 with confusion matrices in Figure 2.13 and a box plot in Figure 2.14. Both VGG-19 and AlexNet placed the vast majority of exemplars in the first category (city car). Likely this was due to all output scores from both networks being identical for each class with the maximum selection (Eqn. 2.2) choosing the first value that is maximum (shown by the recall for the city class being 1.0 for VGG-19 and 0.997 for AlexNet) - as the training was balanced across classes, it is unlikely that this class was optimised for one particular class to achieve a lower score. The ResNet50 was able to get the vast majority of exemplars into the correct class, however still had issues with a number of classes.

The largest confusion for the ResNet50 model is with the people carrier and large hatchback class - an understandable confusion as there is a large similarity between the classes (average images of the two classes can be found in Figure 2.1, with large hatchback row 2 column 1, and people carrier row 2, column 2). This corresponds to a low precision for people carrier (0.69) and for large hatchback a low recall (0.759) and precision (0.864). The city class had a more minor confusion with the hatchback class, although this only affected the precision of the city class (0.796). Otherwise, all per-class scores are above 0.88 and close to 0.9 or above - with the hatchback class achieving the highest f1-score (0.956). Overall the ResNet50 model achieved a 0.933 f1-score, with a precision of 0.936 and a recall of 0.932.

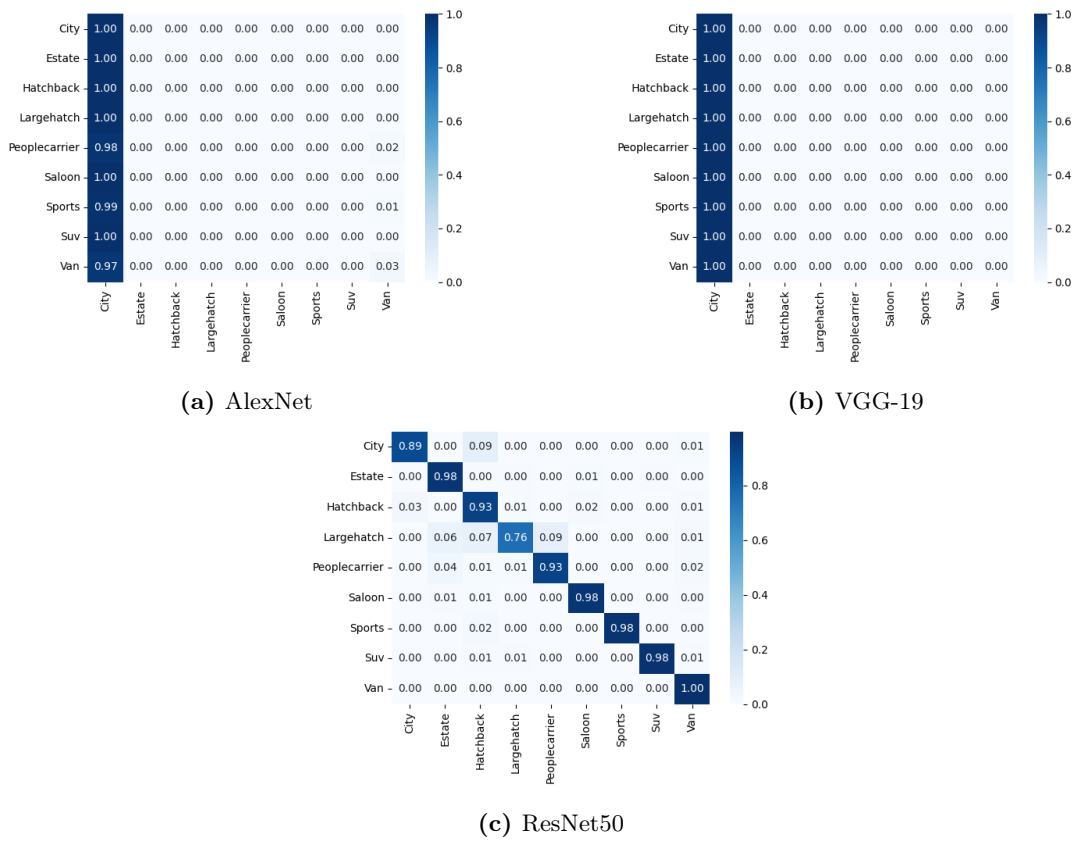


Figure 2.13: Confusion Matrices for Siamese Vehicle Sub-Type Classifiers on Testing Set

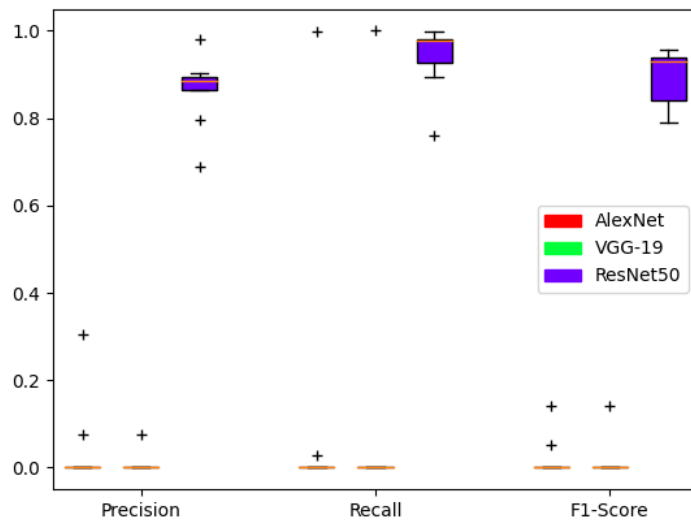


Figure 2.14: Boxplot of class statistics for Siamese Vehicle Sub-Type Classifiers. '+' indicates outliers

Model	Stat	Min	Max	Median
AlexNet	<i>P</i>	0	0.306	0
	<i>R</i>	0	0.997	0
	<i>F1</i>	0	0.142	0
VGG-19	<i>P</i>	0	0.0764	0
	<i>R</i>	0	1	0
	<i>F1</i>	0	0.142	0
ResNet50	<i>P</i>	0.69	0.979	0.886
	<i>R</i>	0.759	0.997	0.977
	<i>F1</i>	0.791	0.956	0.930
HoG-RBF-SVM [21]	<i>P</i>	0.950	0.963	0.958
	<i>R</i>	0.837	0.889	0.866
	<i>F1</i>	0.890	0.924	0.910

Table 2.10: Min/Max/Median of class statistics for Siamese Vehicle Sub-Type Classifiers compared against HoG-RBF-SVM [21]

The AlexNet and VGG-19 models achieved very low scores across the board, with f1-scores of 0.0131 and 0.0109 respectively. In this case, the AlexNet scored slightly higher overall due to having some exemplars from the van class correct - however, this does not represent a statistically significant difference overall as only a few exemplars would cause such a difference.

Table 2.10 shows the min/max/median of the classification statistics alongside the best classifier from [21]. The max/median recall/precision for ResNet50 is significantly better than the HoG-RBF-SVM - however, has a much smaller range in values. The overall weighted average f1-score for ResNet50 (0.933) is however higher than the SVM (0.910).

Figure 2.15 shows three examples of mis-classifications for each class - sports and van have less than three mis-classifications overall so therefore shows all of them. The normalisation process (resizing to a set size) has lost the overall size information of the vehicle. This can be seen in the results for examples such as city \rightarrow SUV or hatchback. In the case of the sports car, the mis-classified images happen to contain a vehicle parked behind them causing confusion to the actual class. In the case of the van, it has identified a mis-labelling within the original dataset.

The lack of performance for the AlexNet model can be explained by its performance in auto-encoder training. Figures 2.7d, 2.8d, 2.9d and 2.10d show that it was not capable of learning a suitable feature encoding to reproduce the initial input imagery - only producing a similar image for every exemplar. Therefore, the AlexNet model once trained has very little information available in the feature encoding stage to determine a significant difference between classes within the overall Siamese network.

An explanation behind the VGG-19 model’s lack of performance is more difficult to determine.

Actual Class	Exemplar 1	Exemplar 2	Exemplar 3
City	 Hatchback	 SUV	 Van
Estate	 Large Hatchback	 People Carrier	 Saloon
Hatchback	 City	 Large Hatchback	 Saloon
Large Hatchback	 Estate	 Hatchback	 People Carrier
People Carrier	 Estate	 Hatchback	 Van
Saloon	 Estate	 People Carrier	 Van
Sports Car	 Hatchback	 Hatchback	— —
Van	 City	— —	— —

Figure 2.15: Vehicle ResNet50 Sub-Type Mis-Classification Examples

Figures 2.7e, 2.8e, 2.9e and 2.10e show that the auto-encoder for this model is capable of learning an encoding to produce an output similar to the original exemplar, unlike AlexNet - however, its performance is worse than the ResNet50 in that it is less capable of reproducing the original colour. While more information is preserved in this encoding compared to the AlexNet model, there is not enough to achieve adequate class separation in the overall Siamese configuration.

2.4.5 Make/Model Results

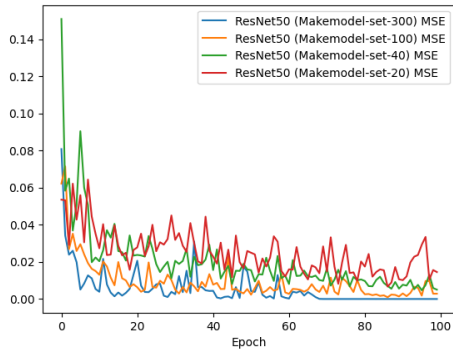
For the Make/Model tests, only ResNet50 was evaluated due to the lack of performance in the sub-type scenario for the AlexNet and VGG-19 architectures. Four separate models were trained on each of the sets provided in the Reading Side-Profile Vehicle Dataset - sets 300, 100, 40, and 20. The MSEs from training can be found in Table 2.11 with graphs showing the training process in Figure 2.16. The trained models' MSEs are ordered with the set containing the fewest classes (Set 300) obtaining the lowest score of 0.00000539, down to the set containing most classes (Set 20) having the highest score of 0.0106 - this is an expected result due to each set being subsequently a harder problem to solve due to the larger number of classes. This observation however does not transfer to the testing set MSEs with Set 100 obtaining the lowest MSE.

Graphs of the classification statistics (precision, recall, and f1-score) can be found in Figure 2.17. The large amount of oscillation while training can be seen here with the set containing the smallest class size (Set 20) having the visibly largest amount. It is worth noting that the testing sets graphs in the figure are only every five epochs, and that the overall range of the graph is within approximately a 0.045 region. For the set 30 model, it can be seen that it trains initially rather quickly before eventually setting down around epoch 70, while set 100 has a period of stability around epoch 80, the other sets never achieve stability.

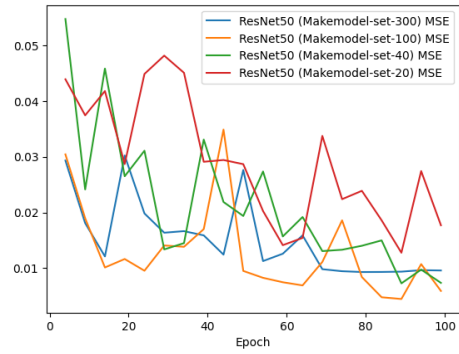
Figure 2.17 shows the calculation of precision, recall, and f1-score throughout the training process. As the number of classes increase (as the number of the set decreases), the overall scores decrease. Set 300 was capable of reaching near 1.0 scores within the first 10 epochs on the training set and near 0.95 on the testing set. Set 100 was able to reach approximately 0.95 on the training set, and approximately 0.90 for the testing set. Set 40 was also able to reach the similar 0.95 score on the training set, although a bit lower, and Set 20 only reaching a 0.80

Dataset	Time to Train	Train Set MSE	Test Set MSE
Set 300	2h 51m 58s	0.00000539	0.00931
Set 100	19h 41m 64s	0.00139	0.00446
Set 40	2d 5h 44m 18s	0.00559	0.00727
Set 20	5d 2m 39s	0.0106	0.0128

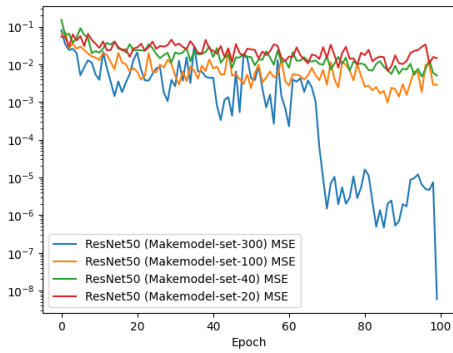
Table 2.11: MSE results of ResNet50 on Vehicle Make/Model Datasets



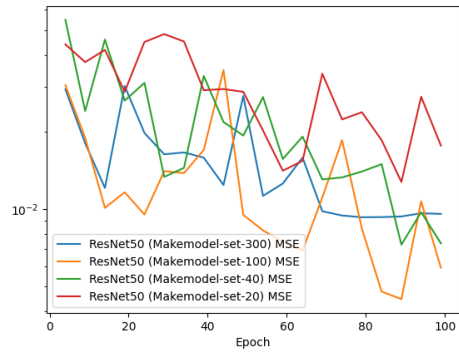
(a) Training Sets



(b) Testing Sets

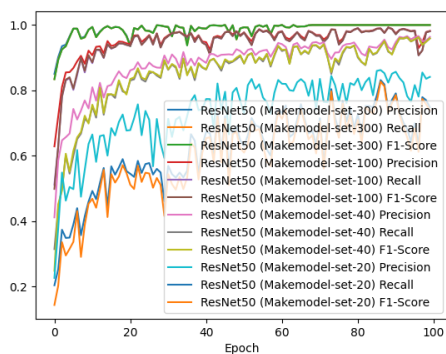


(c) Training Sets (Log)

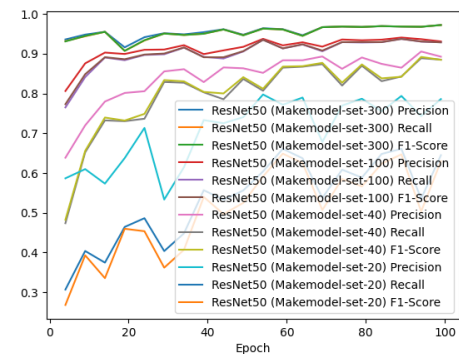


(d) Testing Sets (Log)

Figure 2.16: Epoch Training MSEs for ResNet50 on Vehicle Make/Model Datasets



(a) Training Sets



(b) Testing Sets

Figure 2.17: Epoch Classification Statistics (Precision, Recall, and F1-Score) for ResNet50 on Vehicle Make/Model Datasets

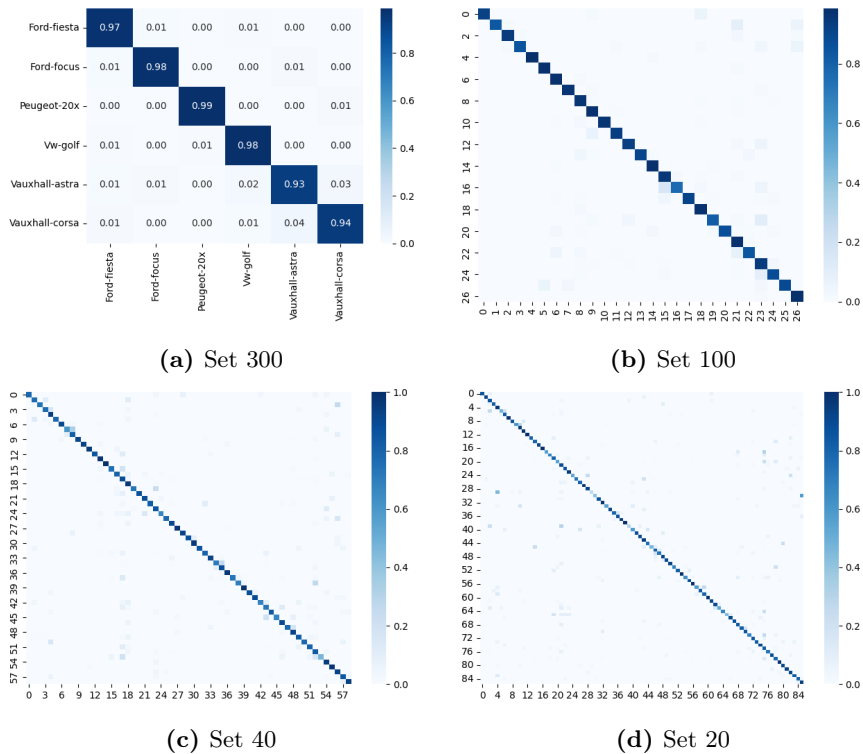


Figure 2.18: Confusion matrices for ResNet50 on testing sets of all make/model sets. Set 100 - set 20 contain no class names due to the larger number of classes

score.

The precision and recall for both sets 300 and 100 are rather similar - however, for both Sets 40 and 20 the precision is significantly higher than the recall. The method of selecting exemplars for training can be found in Section 2.3.1; this is the method of selecting one negative exemplar per one positive exemplar, as well as a random sampling based on the class with the smallest amount of samples. These were an attempt at addressing the issue of balancing precision and recall, however overall this did not confer any benefit. A likely explanation is that there were not enough negative exemplars to properly allow the optimisation algorithm to discriminate against all classes. In the sets with a smaller number of classes (Sets 300 and 100) this isn't a problem, as on average each negative exemplar appears in the training every few epochs due to the low amount of classes. With the sets that contain a larger amount of classes, the likelihood decreases substantially - for example, Set 20 contains 86 classes and over 100 epochs with a random selection (only a few of those class comparisons will be selected multiple times). This likely wouldn't be solved with a larger amount of epochs either, as there is a substantial amount of time before seeing the same negative exemplar again.

Figure 2.18 shows the confusion matrices of all the make/model sets. All sets have a clear strong diagonal line, with only set 20 having a substantial amount of classes resulting in mis-

classifications.

Table 2.12 compares the ResNet50 model with the other RF and SVM models. The best results for the ResNet50 model is for Set 300 which has the second-best min/max/median f1 scores (0.934/0.987/0.974) with HoG-Lin-SVM having the best (0.972/0.989/0.982). While the maximum score stays consistently high across the other datasets, achieving 100% accuracy for some classes, the minimum consistently drops as more classes are introduced.

Table 2.13 shows the top 10 and bottom 10 classes by f1-score, as well as the classes with top and lowest support (the number of exemplars in the testing set). The classes with the best results have overall a relatively low number of exemplars in the testing set, with the top 9 classes having 13-31 exemplars. However, there are also 6 classes in this range in the bottom 10 classes. From the Set 20 results tables, there appear to be little correlation between the number of exemplars and which classes did well and which did badly.

Part of the disparity could come from a weakness in the labelling of the dataset - in that the categories do not differentiate by year of the vehicle. Newer vehicles in the same class can have a very different look than older vehicles. Figure 2.19 shows two mis-classification exemplars for each of the worst 5 classes. In cases like the VW-Polo Exemplar 1 and Honda-Accord Exemplar 2, the older version of the vehicle looks somewhat different to the averaged image due to the change in overall shape in different year models. However this is not the case for all classes within the dataset. Both Honda-CR-V exemplars look very similar to the Volvo-X in the average image, and the Mazda-6 has an overall similar shape to the two compared classes.

Comparing the classes trained in Set 300 (Table 2.14) with the Set 20, it is clear that when trained on a smaller number of classes it is able to perform much better - despite the classes having a large variety in looks. This is likely due to the balanced training method taking a number of exemplars from each class based on the smallest class size. There are far more exemplars to process in Set 300, therefore the training method sees the exemplars more frequently than in Set 20.

The lack of performance with sets of larger number of classes could also be due to the overall larger number of classes to compare against with regards to similarity. The higher the number of classes to compare against allows for the potential for more confusion. While SVMs are able to scale to a larger number of classes, this could be a limitation of the method for a Siamese network classifier using average images of classes.

	Set 20			Set 40			Set 100			Set 300		
	Min	Max	Med	Min	Max	Med	Min	Max	Med	Min	Max	Med
ResNet50	<i>P</i>	0.281	1	0.877	0.45	0.914	0.794	1	0.943	0.934	0.987	0.969
	<i>R</i>	0.27	1	0.854	0.471	0.864	0.766	0.986	0.938	0.934	0.987	0.975
	<i>F1</i>	0.381	1	0.821	0.581	0.873	0.843	0.993	0.937	0.934	0.987	0.974
Img-Forest	<i>P</i>	0.737	0.817	0.782	0.794	0.812	0.825	0.861	0.845	0.872	0.912	0.891
	<i>R</i>	0.57	0.623	0.602	0.675	0.698	0.766	0.797	0.782	0.879	0.912	0.897
	<i>F1</i>	0.643	0.707	0.68	0.73	0.751	0.794	0.828	0.812	0.875	0.912	0.894
Evo-Forest	<i>P</i>	0.843	0.883	0.858	0.829	0.849	0.866	0.908	0.887	0.903	0.938	0.924
	<i>R</i>	0.895	0.922	0.907	0.883	0.904	0.895	0.92	0.911	0.911	0.94	0.927
	<i>F1</i>	0.868	0.902	0.882	0.855	0.893	0.871	0.883	0.899	0.907	0.939	0.925
HoG-Forest	<i>P</i>	0.83	0.847	0.843	0.833	0.85	0.823	0.86	0.854	0.613	0.625	0.616
	<i>R</i>	0.843	0.857	0.851	0.875	0.887	0.949	0.956	0.953	0.977	0.983	0.979
	<i>F1</i>	0.836	0.852	0.847	0.853	0.873	0.865	0.882	0.901	0.753	0.764	0.756
HoG-Lin-SVM	<i>P</i>	0.943	0.961	0.953	0.942	0.95	0.967	0.975	0.97	0.972	0.989	0.982
	<i>R</i>	0.933	0.953	0.947	0.936	0.949	0.966	0.981	0.975	0.973	0.99	0.983
	<i>F1</i>	0.938	0.957	0.95	0.939	0.949	0.966	0.978	0.972	0.972	0.989	0.982
HoG-RBF-SVM	<i>P</i>	0.937	0.956	0.949	0.942	0.948	0.959	0.973	0.969	0.911	0.94	0.924
	<i>R</i>	0.929	0.948	0.937	0.939	0.945	0.967	0.98	0.974	0.881	0.921	0.899
	<i>F1</i>	0.933	0.952	0.943	0.94	0.946	0.963	0.976	0.971	0.896	0.93	0.911

Table 2.12: Min/Max/Median of class statistics for ResNet50 Siamese vehicle make/model classifiers compared with random forest and support vector machine models

Class	P	R	F1	Support
Audi-TT	1	1	1	24
Fiat-500	1	1	1	28
MG-MG-TF	1	1	1	22
Saab-9-5-Estate	1	1	1	13
Citroen-C3	0.969	1	0.984	31
Ford-Puma	0.938	1	0.968	15
Renault-Laguna-Estate	0.933	1	0.966	14
LTC-all	1	0.923	0.96	13
LandRover-RangeRover	0.955	0.955	0.955	22
BMW-Mini	0.914	0.993	0.952	139

(a) Top 10 Classes

Class	P	R	F1	Support
Mazda-3	0.5	0.308	0.381	13
Honda-Accord	0.909	0.27	0.417	37
VW-Polo	0.281	0.832	0.42	101
Mazda-6	0.5	0.583	0.538	12
Honda-CR-V	0.889	0.421	0.571	19
Audi-A4-Avant	0.789	0.484	0.6	31
Peugeot-10x	0.905	0.463	0.613	41
BMW-5	0.457	0.955	0.618	22
Skoda-Fabia-Estate	1	0.455	0.625	11
Fiat-Bravo	0.75	0.562	0.643	16

(c) Bottom 10 Classes

Class	P	R	F1	Support
Ford-Fiesta	0.907	0.543	0.68	519
Ford-Focus	0.879	0.889	0.884	441
VW-Golf	0.855	0.848	0.852	264
Vauxhall-Astra	0.866	0.726	0.79	259
Vauxhall-Corsa	0.654	0.789	0.715	242
Peugeot-20x	0.808	0.818	0.813	154
Ford-Ka	1	0.738	0.849	141
BMW-Mini	0.914	0.993	0.952	139
Toyota-Yaris	0.962	0.857	0.907	119
Renault-Clio	0.963	0.696	0.808	112

(b) Highest 10 Support Classes

Class	P	R	F1	Support
Renault-Grand-Scenic	0.818	0.818	0.818	11
Skoda-Fabia-Estate	1	0.455	0.625	11
Mazda-6	0.5	0.583	0.538	12
Rover-25	0.632	1	0.774	12
LTC-all	1	0.923	0.96	13
Mazda-3	0.5	0.308	0.381	13
Nissan-Primera	0.667	0.769	0.714	13
Peugeot-40x-sw	0.75	0.923	0.828	13
Saab-9-5-Estate	1	1	1	13
Peugeot-30x-cc	0.65	0.929	0.765	14

(d) Lowest 10 Support Classes

Table 2.13: Top/Bottom 10 Make/Model Set 20 Classes for ResNet50

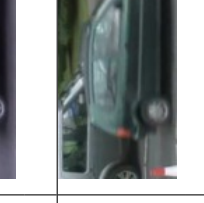
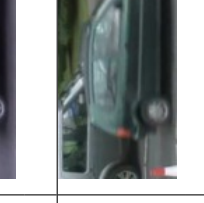



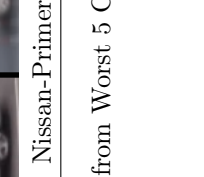
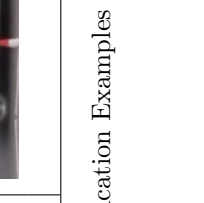

Actual Class	Exemplar 1	Exemplar 2
Mazda-3	 Audi-A3  Audi-A3	 Honda-Civic  Honda-Civic
Honda-Accord	 Audi-A4  Audi-A4	 BMW-5  BMW-5
VW-Polo	 BMW-Mini  BMW-Mini	 Vauxhall-Corsa  Vauxhall-Corsa
Honda-CR-V	 Volvo-X  Volvo-X	 Volvo-X  Volvo-X
Mazda-6	 Ford-Mondeo  Ford-Mondeo	 Nissan-Primera  Nissan-Primera

Figure 2.19: Vehicle ResNet50 Make/Model Set 20 Mis-classification Examples from Worst 5 Classes

Class	P	R	$F1$	Support
Ford-Fiesta	0.979	0.973	0.976	519
Ford-Focus	0.98	0.977	0.978	441
Peugeot-20x	0.987	0.987	0.987	154
VW-Golf	0.959	0.985	0.972	264
Vauxhall-Astra	0.934	0.934	0.934	259
Vauxhall-Corsa	0.954	0.942	0.948	242

Table 2.14: Set 300 Classes Scores for ResNet50

2.5 Conclusions

The Siamese ResNet50 model was able to train and provide the best scores against other methods when trained for the sub-type dataset, and second place with comparable scores for the set 300 make/model dataset. This shows that the Siamese ResNet50 is capable of performing well with a small number of classes (9 for sub-type, 6 for set 300). Even when trained on datasets with a larger number of classes, for the majority of classes it is able to train with a set 20 f1-score median of 0.82 - however it ultimately falls behind SVM models trained on simple HoG features.

The Siamese VGG-19 and AlexNet models were unable to train well in the classification scenarios - even after splitting the network into parts to train the networks separately. It is clear that AlexNet would not perform well after training the auto-encoder, as it was only capable of producing a similar image for every exemplar regardless of class. The VGG-19 model was able to train better than the AlexNet during the auto-encoder stage producing relatively similar imagery to its input, however was ultimately unable to train to be a classifier.

These results show that the advancements of the residual neural network architecture was capable of improving the overall output enough to get good results for a small number of classes, however a limitation of the methodology falls short of being able to train the networks to compete with the SVM models.

Chapter 3

Trajectory Pattern Extraction and Classification

Several computer vision tasks [26]–[29], [60]–[64], including that of extracting dominant motion patterns of moving targets in a scene [30], [65]–[67], rely on the analysis of object trajectories - these are just some of the many studies on the topic. The extracted motion patterns corresponding to motion of targets (trajectories) in a scene offer useful information that could aid in performing activity analysis [22], behaviour prediction [23], tracking [24], abnormality detection [25], [26] and trajectory prediction [27]–[29].

In the context of land border surveillance: analysis of trajectories of individuals or vehicles can be used to alert abnormal behaviours to security officials when a target’s trajectory does not match closely to a known path, or matches a known path that such targets should not be traversing. Clustering is one approach that can be used to identify motion patterns within a scene from a set of trajectories [30], [66], [68]. This has been used as part of an analysis to determine a trajectory auto-encoder deep network architecture [30].

This chapter describes a complete solution for identifying the most suitable deep neural network architecture based on the McCulloch-Pitts neuron [5], as well as using that architecture for the basis of a Siamese Network. The proposed approach assumes that tracking has already been performed to an acceptable level, and uses the datasets from Section 3.2.2 for that purpose. The network looks at the entirety of a trajectory for the purpose of identifying the overall pattern of the trajectory and assigning a cluster/class ID.

This chapter first provides an overview of related work in Section 3.1; next the problem to be solved is defined in Section 3.2; then there is an analysis to determine a neural network architecture in Section 3.3; moving on to the architecture of the entire proposed network in Section 3.4; and the results and analysis can be found in Section 3.6.

3.1 Background

Trajectory applications of Siamese networks are primarily based at the tracklet stage to reinforce tracking. One use has been a part of a temporal analysis on tracklets as part of an overall visual analysis to disentangle object occlusion [24]. Other types of deep learning networks to analyse trajectories have also been used. Auto-encoders have been used to create a trajectory feature vector for clustering analysis [30]. LSTMs have also been used as well as part of determining a trajectory feature [28] and are quite a suitable option considering the varying length of trajectories. Another LSTM-based method to overcome the issue of varying length of trajectories is to use a sequence to sequence (Seq2Seq) auto-encoder to create a feature vector [69]. Trajectory prediction also makes use of LSTMs [27], [28].

While an LSTM seems a natural fit for trajectories due to their varying size, it has been shown that if the full trajectory is already known that fully connected layers can be comparative or better depending on the dataset [30].

There are also approaches outside deep learning and neural networks to perform trajectory analysis. Techniques such as Dirichlet process mixture model [65] and discrete wave transforms [66] have been used. These non deep-learning features are generally out-performed by [30], [69].

3.2 Problem Definition

This section contains descriptions of the mathematical definitions in Section 3.2.1 as well as the datasets used for evaluation of the method in Section 3.2.2.

3.2.1 Mathematical Definition

Let \mathcal{X} be a set of trajectories estimated by a tracker in a video sequence, $V: \mathcal{X} = \{\mathfrak{X}_j\}_{j=1}^J$ where J is the number of estimated trajectories. \mathfrak{X}_j is the estimated trajectory for target j : $\mathfrak{X}_j = (X_{k,j})_{k=k_{start}^j}^{k_{end}^j}$, where k_{start}^j and k_{end}^j are the first and final frame numbers of \mathfrak{X}_j , respectively. $X_{k,j}$ is the estimated state of target j at frame $k: k = 1, \dots, K$ with K as the total number of frames in V . $X_{k,j} = (x_{k,j}, y_{k,j})$, where $(x_{k,j}, y_{k,j})$ denotes at frame k the position of target j on the image plane. The analysis of trajectories (\mathcal{X}) aid in identifying the path (start, approximate movement, destination) of moving objects (people, vehicles) in a scene.

The overall objective of the problem is to assign an input trajectory \mathfrak{X}_j to a class/cluster C_n so that it can be used for further analysis. A representative trajectory belonging to the class/cluster $\mathfrak{X}_j \in C_n$ can be used as a comparison during the training/assignment process, or an average of all trajectories within the class/cluster $\bar{\mathfrak{X}} \in C_n$. A class/cluster C_n can be

defined in a few ways: manual labelling of a known set of trajectories into manually defined classes/clusters, or automated labelling based on clustering approaches such as [30].

3.2.2 Datasets

In order to show the effectiveness of the different approaches, a total of 4 datasets were used with a summary of their information found in Table 3.1. All datasets provide real trajectories generated by the original authors of each dataset, with induced camera motion already compensated for.

The first two are related to traffic monitoring and called Traffic Junction and Parking Lot [66] respectively. Traffic Junction offers a busy junction scenario with vehicles moving in varying directions as well as people walking across and alongside roads. Parking lot presents a multi-row car park scenario with vehicles and people targets. Both Traffic Junction and Parking Lot are recorded from a mobile aerial platform. Real trajectories are used with induced camera motion already compensated for both datasets as provided by the original authors.

For further analysis of the proposed approach, two non-traffic datasets (Students003 [70] and Train Station [23]) were also evaluated. Students003 offers a highly crowded scene with people walking around in an outdoor open area; Train Station offers an open area that is highly crowded with people moving inside a train station. Both Students003 and Train Station are recorded from a top-down(ish) fixed camera. On Students003 and Train Station we use the available trajectories by respective authors [23], [70]. Only longer trajectories (length > 600) [66] for the Train Station dataset are used due to the large amount of shorter trajectories (or tracklets) within the published dataset that do not fit into any particular over-arching motion or pattern.

Representative images and trajectories of all the datasets can be seen in Figure 3.1, a summary of the original videos from the datasets can be found in Table 3.1, and information on the trajectories from each dataset can be found in Table 3.2.

Dataset	Frame Size	Frame Count	Trajectory Count	FPS
Traffic Junction	540 x 960	16154	162	30
Parking Lot	1080 x 1920	9517	37	30
Students003	576 x 720	5405	301	25
Train Station	480 x 720	46009	379	23

Table 3.1: Summary of all trajectory datasets’ video structures

Trajectories are pre-labelled into clusters that are primarily based on the approximate (x, y) co-ordinates where a trajectory enters and exits the scene, for instance, the start and end of a traffic lane when entering/exiting the scene. In the Parking Lot scenario, this also includes

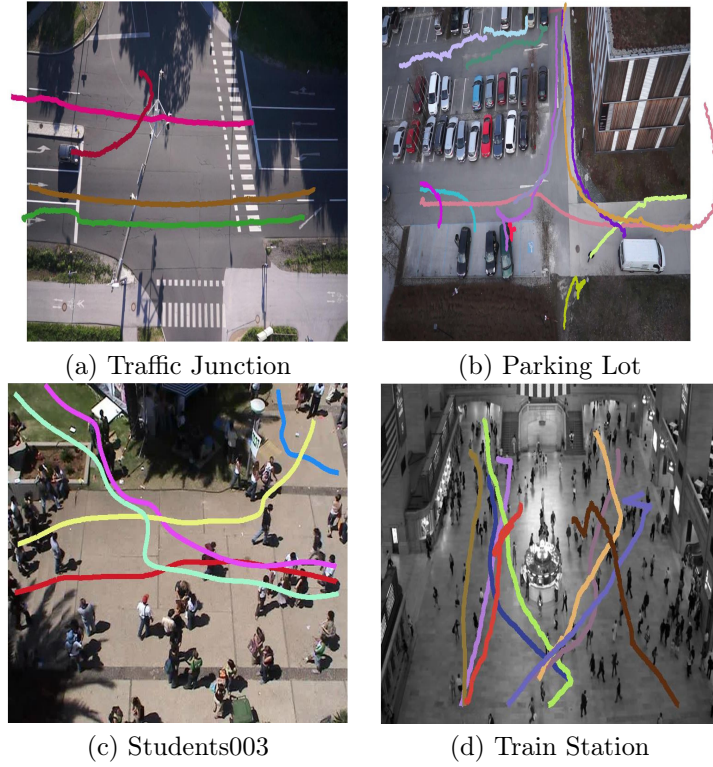


Figure 3.1: Representative Images and Trajectories of the Datasets used for Evaluation. (a) and (b) are from [66], with (c) from [70] and (d) from [23].

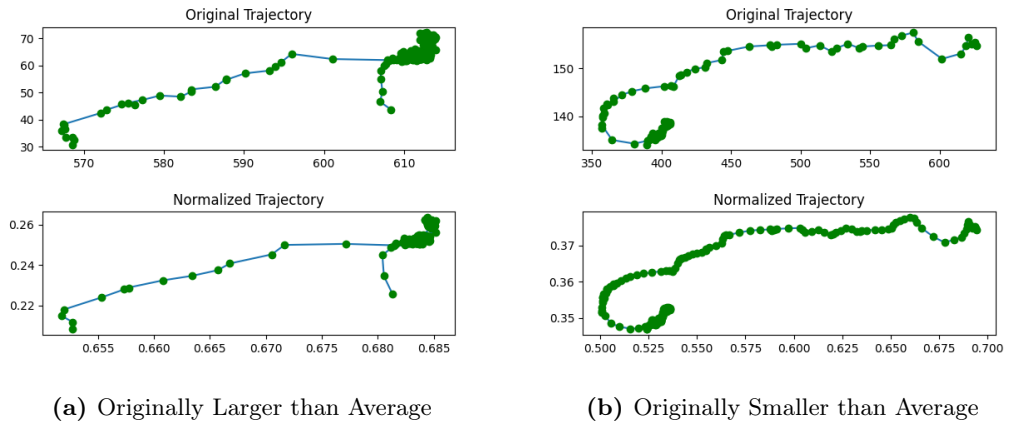


Figure 3.2: Trajectories Before and After Normalisation from the Traffic Junction Dataset

Dataset	Min Length	Max Length	Mean Length
Traffic Junction	17	1606	179.28
Parking Lot	29	2622	545.98
Students003	37	2873	497.13
Train Station	601	1512	731.42

Table 3.2: Dataset Trajectory Information

vehicles observed parking into and out of a parking slot, without starting/finishing at the edge of the scenario.

The clusters are defined to be omnidirectional i.e. trajectories entering/leaving on opposite sides with similar positioning will be placed into the same cluster. In the vehicle scenarios, this omnidirectional property largely does not have an effect due to vehicles driving through lanes within the scene - however, the pedestrian scenarios have individuals walking along paths in opposing directions. This adds a challenge for feature extractors as it requires them to produce a feature vector agnostic of the direction of the trajectory.

There are multiple ways that one could prepare the datasets for use in a method [71]. Broadly, these come under four categories: transformation, re-sampling, substitute or adding additional features. In this paper, all datasets are prepared using a re-sampling methodology to achieve a static overall size per dataset based on the mean average trajectory length of its dataset. The x and y co-ordinates are also normalised between 0 and 1 based in the minimum and maximum of the dataset's original co-ordinate space. Two examples of the normalisation can be found in Figure 3.2 showing a trajectory originally larger than the mean average, and a trajectory originally smaller than the mean average.

3.3 Auto-Encoder Architecture

This section is split into two parts: Section 3.3.1 describes the overall mathematical model of the architecture, followed by Section 3.3.2 describing the methodology used to design the overall network. The training methodology for the auto-encoder stage is described in Section 3.5.

3.3.1 Mathematical Model

In order to generate the feature vector \mathbf{f}_j for a trajectory \mathcal{X}_j , an auto-encoder (also known as autoassociator or diabolo network) [17] is first trained to reproduce the set of trajectories \mathcal{X} . Once a network has been trained, the output of its smallest layer is used as the feature vector \mathbf{f}_j . Separate network architectures for each dataset have been generated for evaluating each scenario separately using the mean length as described in Table 3.2, as well as a combined network architecture based on all datasets for evaluating the viability of cross-dataset training.

An auto-encoder is an arrangement of a neural network where the output, once trained, is an estimation of the input vector that is provided (Figure 3.3). Outputs of any of the layers in a trained network can be used as a representation of the input, due to the ability of the rest of the network to reproduce the input vector.

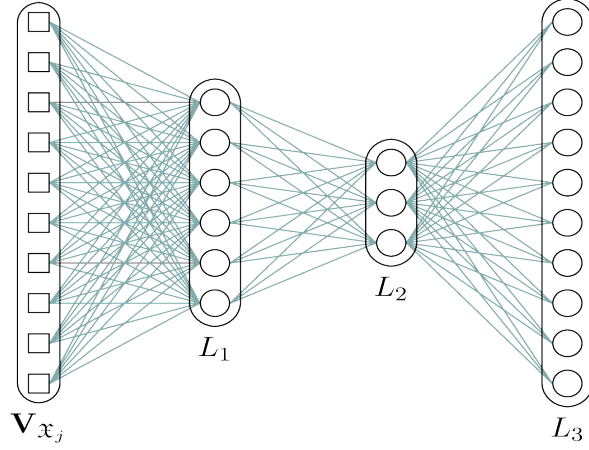


Figure 3.3: Proposed feature representation uses a neural network-based approach that employs the output of the smallest hidden layer (L_2) of a trained auto-encoder to represent trajectory information. $\mathbf{V}_{\mathfrak{x}_j}$: vectorisation of the data of the j th trajectory (\mathfrak{x}_j); L_1 : first hidden layer; L_3 : output layer.

In this trajectory scenario, the input into the network is the vectorisation of the data of \mathfrak{x}_j and is denoted as $\mathbf{V}_{\mathfrak{x}_j}$:

$$\mathbf{V}_{\mathfrak{x}_j} = [x_{k,j}, y_{k,j}]_{k=k_{start}}^{k_{end}^j}. \quad (3.1)$$

A neural network is a combination of small units called neurons that are built up into multiple layers. The type of neuron used in this paper is based on the McCulloch-Pitts neuron [5] - this multiplies a single-dimensional input vector with a weight vector summed with a bias node, and outputs a single value:

$$y_{l,n} = \sum_i^{I_l} (w_{i,n} \mathbf{x}_{l,i}) + b_n, \quad (3.2)$$

where $y_{l,n}$ donates the output pre-activation-function y of the neuron n in layer l . I_l is the length of the input vector X_l for a particular layer, w_i and $\mathbf{x}_{l,i}$ is a value in the weight vector and input vector, respectively, and b is the bias value. The weight vector is initialised to random values. A sigmoidal function is used for the activation function:

$$Y_{l,n} = \frac{1}{1 + e^{-y_{l,n}}}, \quad (3.3)$$

where Y is the output of the neuron.

The neurons are then placed alongside each other as a layer. The output of a layer l can be described by the following equation:

$$L_l = [Y_{l,1}, \dots, Y_{l,I_l}], \quad (3.4)$$

where the layer L_l is a vector containing all the outputs of the neurons in the layer. The next layer is then provided with the output of the previous as its input vector (known as a fully-connected layer), excluding the first layer ($l = 1$) for which the input vector is the vectorised trajectory $\mathbf{V}_{\mathbf{x}_j}$ rather than a previous layer:

$$\mathbf{X}_l = \begin{cases} \mathbf{V}_{\mathbf{x}_j}, & \text{if } l = 1; \\ L_{l-1}, & \text{if } l > 1. \end{cases} \quad (3.5)$$

The architecture of this type of network contains two stages: an encoding and decoding stage. For encoding, the number of neurons in each layer decreases so that the dimensionality of the original input vector is effectively reduced when passed through the network. The decoding stage is the opposite: a set of layers incrementing in size up to the original length of the input vector. Both of these stages consist of one or more layers. As mentioned above, this method uses pre-defined scales based on the length of the original feature vector to determine the number of neurons in each layer in the encoding stage. These scales are determined in Section 3.3.2.

3.3.2 Determining the Layered Architecture

It is a hard task to determine an architecture for a novel use; however, in this instance the networks are on the small side as compared to many modern architectures. Training networks of these size takes significantly less time compared with its convolutional counterparts - as such, a simple exhaustive hyperparameter grid search [72] is possible. One of the primary negative attributes of the grid search is that it very quickly becomes an incredibly high dimensional search space due to the exhaustive search of parameters. While another hyperparameter search may be faster, due to the small training time of this type of network, a grid search becomes the optimum choice to determine the best.

The hyperparameters used for the search are described in Table 3.3 along with their training and validation set loss (MSE) on the Traffic Junction dataset. The encoding layers refers to how deep the network is, with the scales a fraction of the original trajectory size. For each depth,

Number of Encoding Layers	Scales	Training Loss	Validation Loss
1	[0.05]	0.0195	0.0234
2	[0.525, 0.05]	0.0146	0.0163
3	[0.6833, 0.3666, 0.05]	0.0174	0.0190
4	[0.7625, 0.525, 0.2875, 0.05]	0.0266	0.0268
5	[0.81, 0.62, 0.43, 0.24, 0.05]	0.0582	0.0561

Table 3.3: Hyperparameters for Trajectory Auto-Encoder Layer Architecture

Dataset	Input \mathbf{V}_x Size	L_1 Size	L_2 Size
Traffic Junction	358	35	17
Parking Lot	1092	109	54
Students003	994	99	49
Train Station	1462	146	73

Table 3.4: The Final Hidden Layer Sizes of Auto-Encoders for Trajectory Datasets

a total of five experiments were performed in order to account for randomness. The final layer was fixed at 5%, with the preceding layers being calculated at equal separation distance from 5% to 100%.

Overall two layers are used due to it having the best results. The first is 10% the length of the vector, the second is 5% of the vector - chosen after further testing on all datasets. Only one layer is used in the decoding stage of the same size as the original vector. These scales are static across all the datasets used in this study. The actual layer sizes are described in Table 3.4.

3.4 Siamese Neural Network Architecture

The overall architecture of the neural network is split into two sections: an encoding section trained as an auto-encoder (architecture described in Section 3.3), and its extension into a twin (or Siamese) neural network. The full architecture can be seen in Figure 3.4. Section 3.5 describes the overall training methodology.

The concept of a Siamese network [1] is to create a network of two parts. An auto-encoder was chosen as the first part as it has been shown one can effectively be used to create a trajectory feature vector to discriminate between clusters [30] and therefore be a good discriminator between trajectories as part of a Siamese network. The network takes two inputs ($\mathbf{V}_{x_j,0}$ and $\mathbf{V}_{x_j,1}$) through the previously trained feature network (excluding the final decoding layer) to produce two outputs, in this case: $L_{2,0}$ and $L_{2,1}$. For the purposes of training and testing, $L_{2,1}$ is an average of all the trajectories in $L_{2,0}$'s class. A depiction of the full network can be seen in Figure 3.4.

The absolute difference of the two vectors is calculated for the input of the Siamese layer:

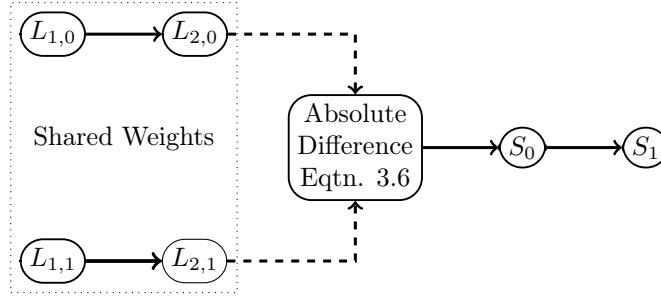


Figure 3.4: Architecture of the full Siamese Trajectory Network. All layers are fully connected layers.

$$D = |L_{2,0} - L_{2,1}| \quad (3.6)$$

The input is then forwarded into a layer equal to the size of L_2 , S_0 , into a single layer of a single neuron S_1 . Both layers use a sigmoidal activation function as in Equation 3.3.

3.5 Training Methodology

There are five datasets that are used for testing. The datasets as described in Section 3.2.2 (Traffic Junction, Parking Lot, Students003, Train Station), as well as a combined dataset (All) that uses all four datasets' trajectories. Each dataset is split into various sub-sets with the following ratios: training 0.4, validation 0.2 and testing 0.4.

There are two stages to training the overall network. An initial pre-training is performed for the encoding section of the network before moving on to train the Siamese network as a whole. Both parts of training use the normalised direction preserving ADAM optimiser (ND-ADAM) [16].

The first stage of training involves training the auto-encoder separately from the rest of the network; it is then trained to reproduce its input. A total of 10 versions of the network are trained for 40000 epochs to account for randomness in the network initialisation at a learning rate of $1e^{-4}$. The network is trained on the Mean Squared Error (MSE) with validation set being monitored at each epoch; the best network is chosen via the best validation score and is chosen at that checkpoint.

The second stage involves training the network as a twin neural network. The final layer of the auto-encoder is removed, and a new layer is added for the similarity output. The layers of the auto-encoder are frozen so no further optimisation is performed, and training occurs for 2000 epochs with a learning rate and weight decay being determined by a Bayesian hyperparameter

Model	Traffic Junction			Parking Lot			Students 003			Train Station		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Traffic Junction	1.00	1.00	1.00	0.73	0.77	0.72	0.66	0.61	0.60	0.53	0.54	0.50
Parking Lot	0.95	0.88	0.90	0.96	0.94	0.94	0.45	0.47	0.43	0.63	0.54	0.50
Students003	0.92	0.85	0.85	0.87	0.77	0.75	0.91	0.91	0.91	0.61	0.51	0.49
Train Station	0.92	0.77	0.81	0.74	0.82	0.77	0.53	0.52	0.50	0.95	0.94	0.94
All (Combined Training)	0.97	0.91	0.93	0.96	0.94	0.94	0.73	0.70	0.69	0.72	0.70	0.70

Table 3.5: Evaluation of the Output of Trajectory Siamese Networks with Models Trained on Each Dataset

optimisation search via Weights and Biases [73]. The only part of training at this stage is on the singular output neuron after the differencing of feature vectors. The training of the overall Siamese network is approached with a one-shot approach [74]: for each epoch, a random trajectory from each class is chosen to be compared to both another of the same class, alongside a random trajectory from another class. This helps against overfitting the network to any one class, as every class gets an equal share of the optimisation. A total of 10 versions are trained with the best validation F1 scores being checked at each epoch; as with the first stage, the best network is chosen via the best validation score and is chosen at that checkpoint.

As mentioned above, both training phases extract out the best state of the network based on the validation set. This is another method to attempt to reduce overfitting to the training set.

3.6 Results and Discussions

This section presents the experimental validation of the methodology by evaluating the Siamese networks results against each other, as well as comparisons to other models. These comparisons use the statistics precision (P), recall (R), and F1 ($F1$) score.

Table 3.5 shows both the results of the trained network on its own dataset, but also the generalisability results of the network on the other tested datasets. Confusion matrices of these results can be found in Figure 3.5. It is clear that the models trained on their own dataset provide the best results, with the best F1 scores for Traffic Junction (1.00), Parking Lot (0.94), Students 003 (0.91) and Train Station (0.94) visible across the diagonal of Table 3.5. The generalisation ability of the models when trained on individual datasets differs substantially, with the performance degradation differing based on the type of dataset. When trained on Traffic Junction and Parking Lot each, the performance deteriorated significantly more for both Train Station and Students003 - this is an expected result as the latter datasets offer substantially different scenarios and challenges.

It is however interesting to note that when trained on Train Station and Students003, the performance degradation is comparatively lesser ($F1 = 0.94$ on Train Station reduces to $F1 =$

Model	P	R	F1
Traffic Junction	0.78	0.66	0.68
Parking Lot	0.73	0.60	0.62
Students003	0.80	0.72	0.73
Train Station	0.81	0.76	0.76
All (Combined Training)	0.79	0.75	0.75

Table 3.6: Overall Scores of Trained Trajectory Siamese Models for all Datasets

0.81 and $F1 = 0.77$ on Traffic Junction and Parking Lot, and $F1 = 0.91$ on Students003 reduces to $F1 = 0.85$ and $F1 = 0.75$ on Traffic Junction and Parking Lot), thus showing a greater generalisation with these models’ training. This could be due to a larger number of similar classes in the Students003 and Train Station datasets - thus the network trains to be better at separating the class clusters. Another point to mention is that these two models have a substantially larger degradation on each other - with the Train Station model ($F1 = 0.94$) reducing to $F1 = 0.50$, and the Students003 model ($F1 = 0.91$) reducing to $F1 = 0.49$. There is a large dissimilarity between the two scenes, with Train Station being an indoor crowded scenario and Students003 showing an outdoor crowded scenario. The two datasets may seem similar at first, however the primary flow of trajectories is on an opposite axis, with Train Station trajectories mostly flowing vertically and Students003 flowing horizontally. Therefore, while the models become good at separating similar clusters, it only learns this fine-grained separation on its particular axis - this can be seen in Figure 3.6.

When training is combined across all datasets, the per-dataset F1 score suffers. The Parking Lot dataset is the only one to have on-par performance with the model that was specifically trained ($F1 = 0.94$ reduces to $F1 = 0.94$), with Traffic Junction at a slightly lower rate ($F1 = 1.00$ reduces to $F1 = 0.93$). The performance deterioration on Students 003 ($F1 = 0.91$ reduces to $F1 = 0.69$) and Train Station ($F1 = 0.94$ reduces to $F1 = 0.70$) is significantly higher, with an overall final score of approximately $F1 = 0.70$ for both.

Table 3.6 describes the overall combined scores across all datasets. Overall the Train Station model takes the top score ($F1 = 0.76$) with the All combined model just slightly behind ($F1 = 0.75$), with the Students003 model taking up 3rd place ($F1 = 0.73$). This confirms that the Train Station and Students003 models from a broader perspective are better at the generalisation than the Traffic Junction and Parking Lot - however, it is worth noting that the latter two datasets have simpler trajectory cluster separation. The All combined model also doesn’t end up at the top of any category of Precision/Recall/F1 Score, even though its lowest score (as seen in Table 3.5) is the highest among all the trained models. An explanation of this disparity can be placed on the number of trajectories in each dataset (Table 3.1) with Students003 (301) and Train

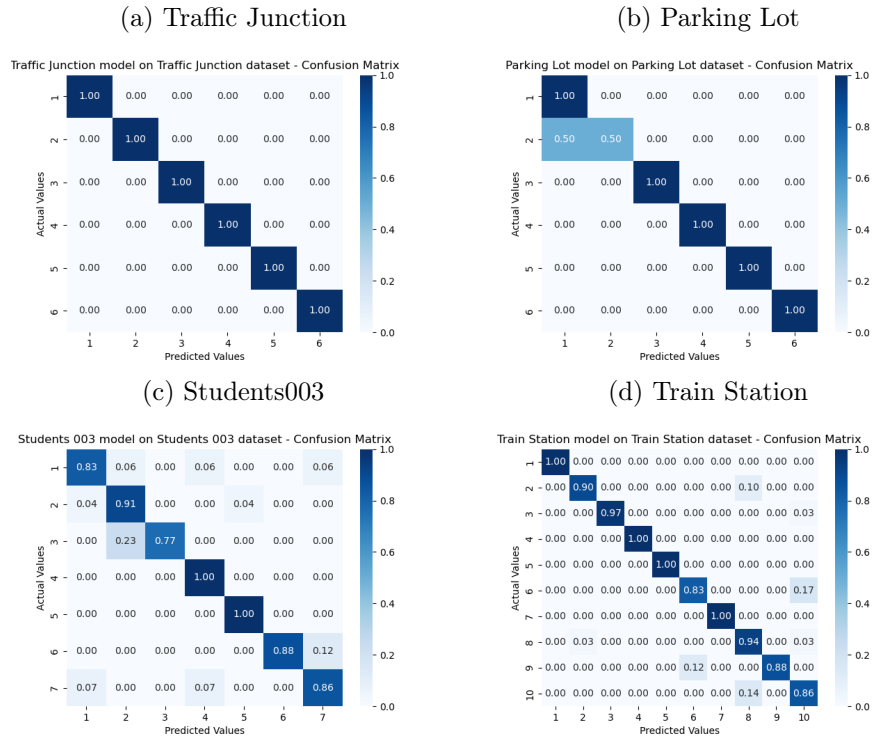


Figure 3.5: Confusion Matrices of Siamese Model on Same Trained Dataset

Station (379) having significantly more trajectories than Traffic Junction (162) and Parking Lot (37). The lower performance of the ‘All’ combined model is likely to be due to the method of training - each trajectory class is given the same priority, therefore the simpler clusters could be prioritised during training as this provides a simpler path to a better score even if it doesn’t end up as the best overall.

The confusion matrices of the specifically trained models can be found in Figure 3.5. All the models have trained relatively well with a strong diagonal with only a few exceptions. Traffic Junction has no confusion with a perfect accuracy; this dataset has well separated clusters with the direction of travel being identical. Parking Lot only has one mis-classification from a cluster 2 being misidentified as a cluster 1; this is likely partially due to the low number of 9 exemplars for this class in the overall dataset - the model appears to have differentiated these trajectories based on direction rather than shape, and the misidentified trajectory is starting from the same position. The Students003 and Train Station models are mostly correct with relatively few mis-classifications, although a number of these are highly visible.

Figure 3.7 shows confusion matrices that are alternate combinations of models trained on one dataset, and then tested on other datasets. Figure 3.7a shows the Traffic Junction trained model on the other three datasets, followed by Figure 3.7b showing the Parking Lot, as well as Figure 3.7c and Figure 3.7d showing Students 003 and Train Station respectively.

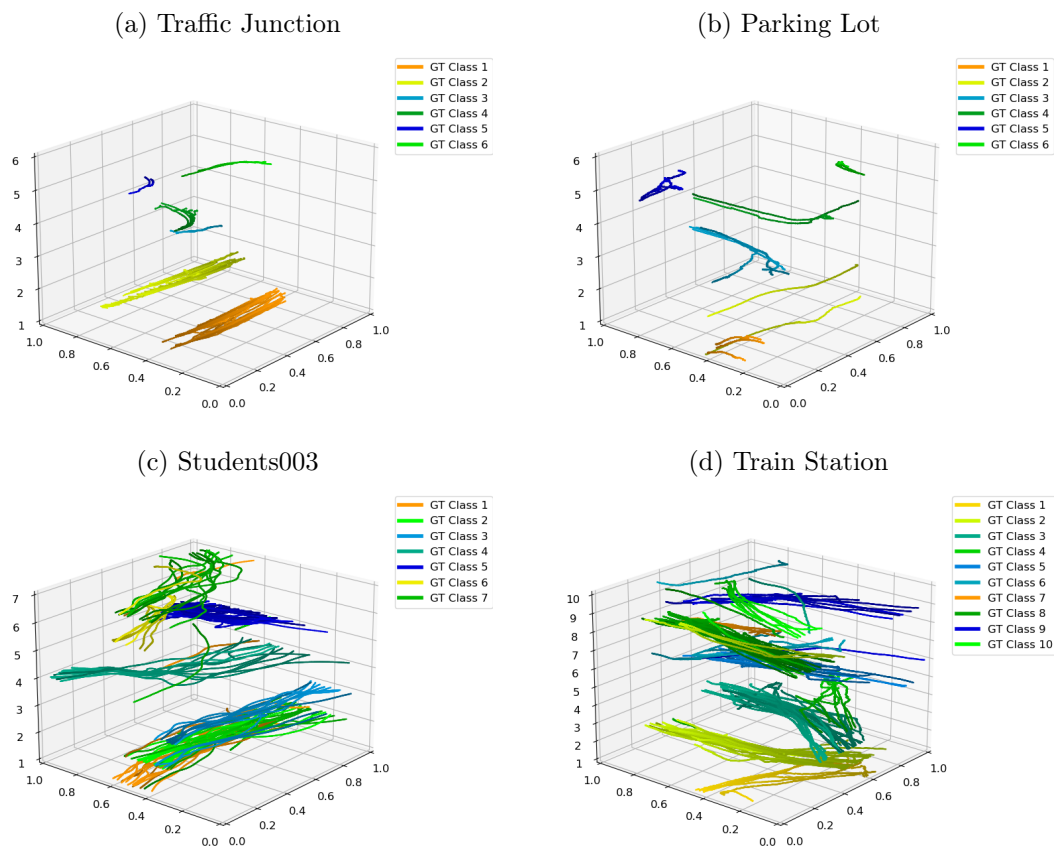
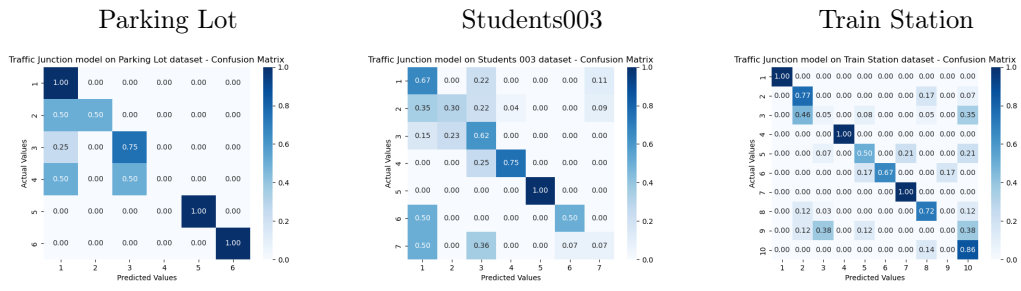
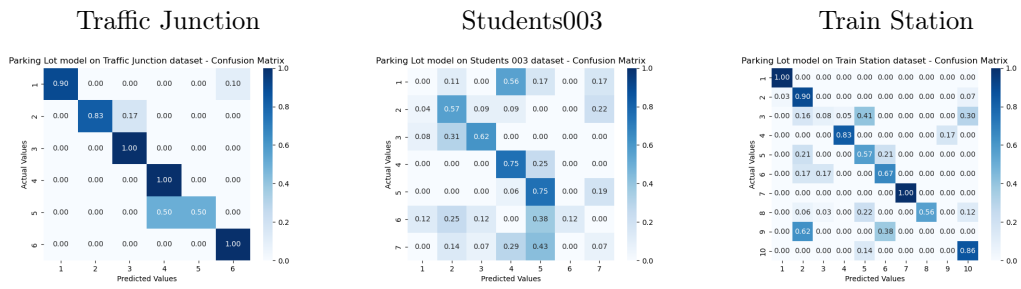


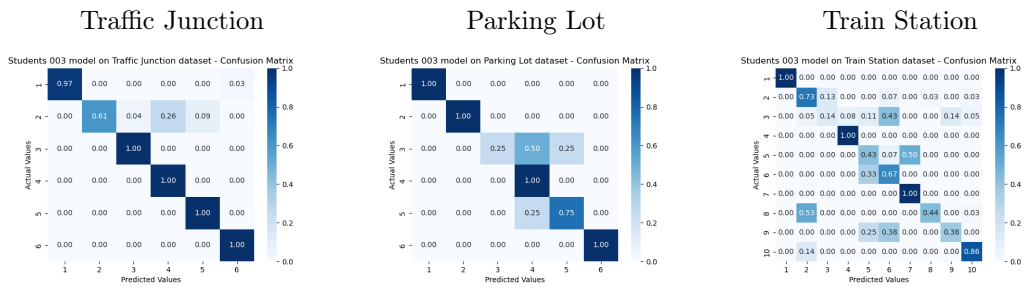
Figure 3.6: Classified trajectories of the Siamese models on the same trained dataset. Each trajectory is shaded with the darker end being the start and the lighter end being the end. The Y-axis shows classified class, and the colour specifies ground truth.



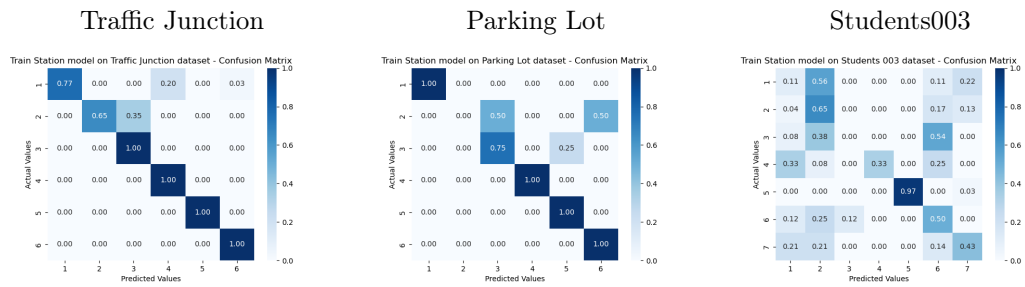
(a) Traffic Junction Model



(b) Parking Lot Model



(c) Students003 Model



(d) Train Station Model

Figure 3.7: Confusion Matrices of Specifically Trained Models Tested on Other Datasets

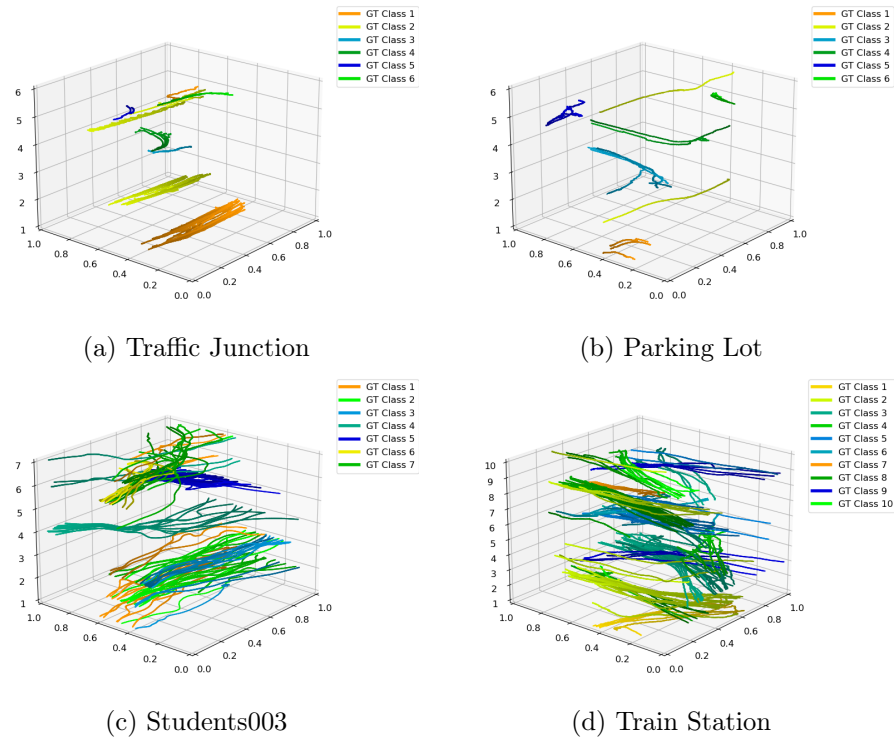


Figure 3.8: Classified Trajectories of Models Trained on All Datasets

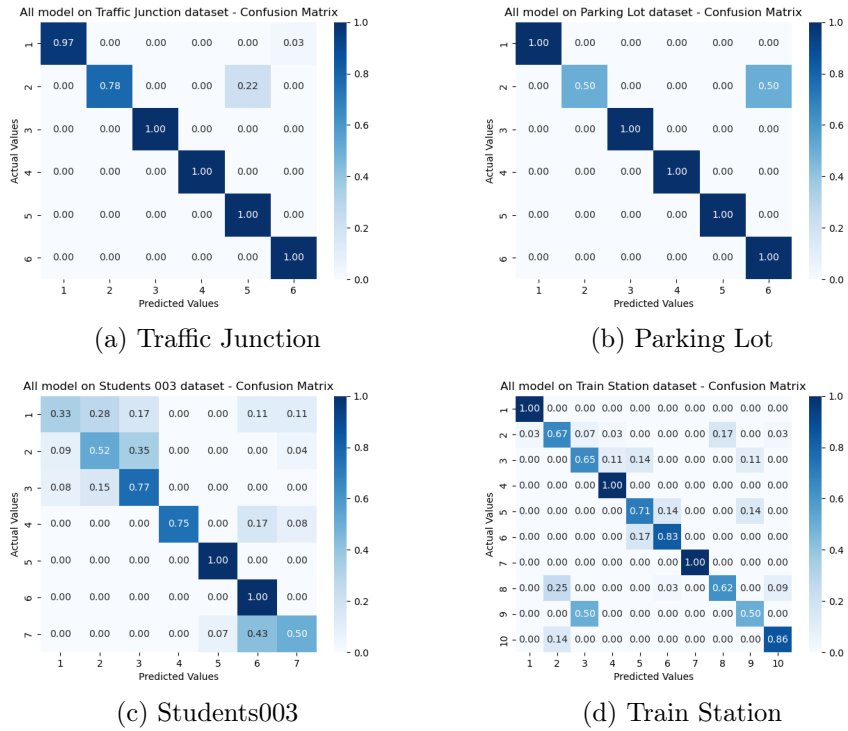


Figure 3.9: Confusion Matrices of Models Trained on All Datasets

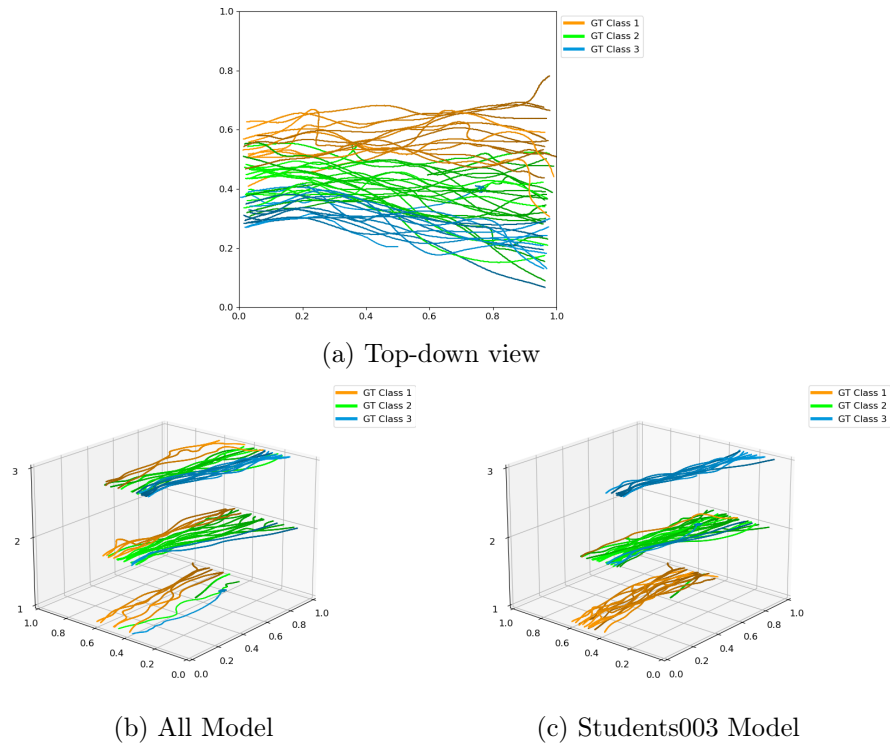
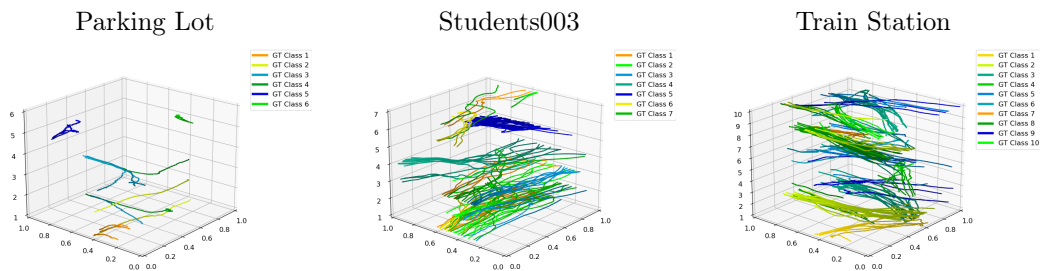


Figure 3.10: Classifications of the model trained on all datasets (b) and the Students003 model (c) for classes 1-3 of Students003. The top (a) shows a top-down 2D view.

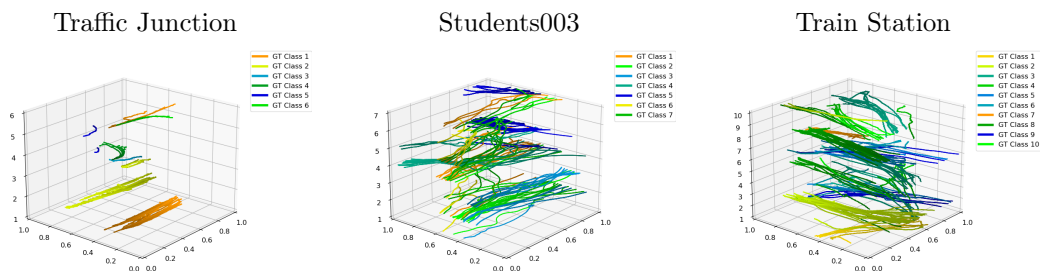
A visualisation of the classified trajectories from the model trained on all datasets can be found in Figure 3.8 with their respective confusion matrices in Figure 3.9. This model only suffers from a few mis-classifications for the Traffic Junction and Parking Lot - however, a large amount of confusion can be seen in the Students003 and Train Station datasets. A large amount of the confusion can be seen in Figure 3.10 for the combined All trained model on Students003 - this also shows how similar the three classes are excluding their vertical alignment and direction of travel with some overlap. It is clear that when trained directly on this dataset that a model is able to determine significantly better the separation between two classes, despite the similarity with only a few mis-classifications on the edges of the class.

Figure 3.11 shows visualisations of the output trajectories that are alternate combinations of models trained on one dataset, and then tested on other datasets. Figure 3.11a shows the Traffic Junction trained model on the other three datasets, followed by Figure 3.11b showing the Parking Lot, as well as Figure 3.11c and Figure 3.11d showing Students 003 and Train Station respectively.

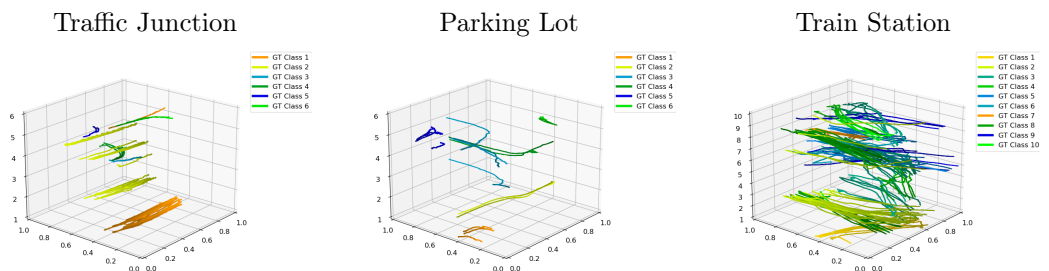
These models have also been compared against state-of-the-art models in Table 3.7. The specifically trained Siamese models results in a tie on one dataset, and is on top in two further sets - with Traffic Junction (Siamese Specific $F1 = 1.00$ -> Movelets $F1 = 1.00$) being equal, Parking



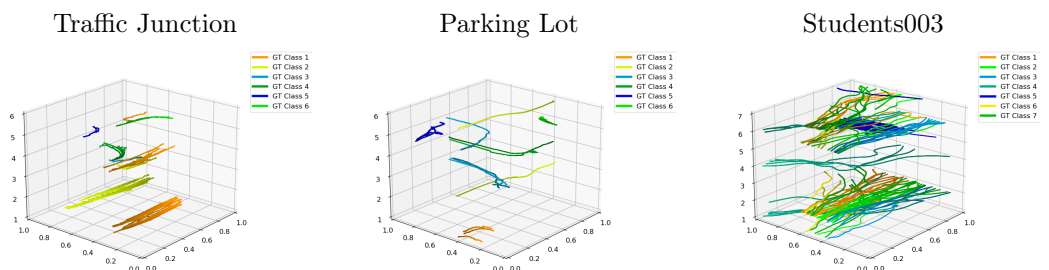
(a) Traffic Junction Model



(b) Parking Lot Model



(c) Students003 Model



(d) Train Station Model

Figure 3.11: Visualised Trajectories of Specifically Trained Siamese Models when Tested on Other Datasets

Dataset		DFTfeat [65]	MULTfeat [75]	DWTfeat [66]	DEEPfeat [30]	Movelets [67]	Siamese Specific [38]	Siamese Combined
Traffic Junction	<i>P</i>	0.67	0.40	0.52	0.80	1.00	1.00	0.97
	<i>R</i>	0.27	0.33	0.50	0.50	1.00	1.00	0.91
	<i>F1</i>	0.38	0.36	0.51	0.61	1.00	1.00	0.93
Parking Lot	<i>P</i>	0.48	0.63	0.65	0.38	0.64	0.96	0.96
	<i>R</i>	0.53	0.33	1.00	0.83	0.71	0.94	0.94
	<i>F1</i>	0.51	0.43	0.79	0.52	0.68	0.94	0.94
Students003	<i>P</i>	0.90	0.60	0.58	0.64	0.96	0.91	0.73
	<i>R</i>	0.40	0.28	0.51	0.60	0.96	0.91	0.70
	<i>F1</i>	0.55	0.38	0.54	0.62	0.96	0.91	0.69
Train Station	<i>P</i>	0.60	0.35	0.45	0.47	0.93	0.95	0.72
	<i>R</i>	0.18	0.18	0.50	0.56	0.89	0.94	0.70
	<i>F1</i>	0.28	0.24	0.47	0.51	0.90	0.94	0.70

Table 3.7: Evaluation results of methods in terms of P , R and $F1$ scores; the higher the score, the better. Top two methods and tied second scores are shown in bold.

Lot is on top (Siamese Specific/Combined $F1 = 0.94 \rightarrow$ DWTfeat $F1 = 0.79$), Students003 is in second place (Movelets $F1 = 0.96 \rightarrow$ Siamese Specific $F1 = 0.94$), and Train Station is on top (Siamese Specific $F1 = 0.94 \rightarrow$ Movelets $F1 = 0.90$). In most cases, the combined model came in third place, excluding the Parking Lot with a joint first place.

3.7 Conclusions

The trajectory Siamese models was able to match and outperform the performance of state-of-the-art models on three out of four datasets when trained on each individual dataset, but fell short when all datasets were included in training a singular model. The generalisation ability of models trained on singular datasets varied depending on the trained dataset - the ones trained on the more complex datasets were able to provide much higher performance on other datasets. This confirms that the overall architecture of the network is performant for these scenarios.

The deficiency of the model trained across all datasets can be attributed to the complexities of certain datasets above others, as well as the training methodology. The one-shot style approach to training was employed in an attempt to balance out the lack of trajectories in certain clusters, however, this ended up being a detriment when training across multiple datasets due to prioritised training for certain classes. Further consideration needs to be considered as to which clusters of trajectories should be the focus on training. It is clear that training on datasets with overall smaller cluster separation provides a model that can separate classes better.

Chapter 4

Differential Face Morphing Detection

Face morphing is the process of combining two facial images into one - creating an image that looks close enough to both original sources. This is a serious security threat for automated facial recognition systems, especially for Automated Border Control (ABC) eGates that verify if the face photo stored in the electronic Machine Travel Document (eMRTD), such as an ePassport, corresponds to the live face of the claimed document owner. The output image of this process not only looks like the two original individuals but has the serious potential to fool facial recognition systems into believing both individuals are the same person [31] - allowing both to pass through the border with the same passport. It was classified as a vulnerability in ISO-19792 (Security Evaluation of Biometrics) [32]. The first simple approach to perform this attack on biometric systems was published by Ferrara et al. [33] - with the first detector being proposed by Raghavendra [34].

There are two types of detector for face morphing detection: S-MAD and D-MAD [76].

- **Single-image Morphing Attack Detection (S-MAD):** A suspected face morph image is checked in order to determine if the image itself is morphed *without* using any prior information.
- **Differential-image Morphing Attack Detection (D-MAD):** A suspected face morph image is compared against a reference image to determine if the suspected image is morphed. For example, a live face image captured in a trusted environment such as an eGate.

This chapter directly addresses the challenge by developing a novel Siamese approach to D-MAD by using multiple common convolutional architectures as a base network, whilst comparing against their performances for S-MAD.

A background of the subject can be found in Section 4.1; a definition of the problem in Section 4.2, including a mathematical definition of the problem in Section 4.2.1, a description of the datasets used in Section 4.2.2, and the evaluation metrics in Section 4.2.3; the model

descriptions can be found in Section 4.3; the methodology can be found in Section 4.4 with a discussion of the results in Section 4.5; and information regarding deployment within the D4FLY project is described in Section 4.6 with a final conclusion in Section 4.7.

4.1 Background

There are numerous S-MAD techniques that have been investigated in the literature [77]. This includes the application of texture analysis with image descriptors as hand-crafted features [78], using Photo Response Non-Uniformity (PRNU) [79] or Scale-Space features [80]. Deep Learning has also been applied for S-MAD using feature-level fusion of Deep CNNs [81]. Standard models such as VGG-19 have also been tested [82]. Layer-wise relevance propagation has been used to analyse the learned features of deep networks for S-MAD [83] and to provide a tool that can highlight traces of forgery in morphed face images [84]. A combination of discrete wave transforms and deep residual learning has been shown to be effective [85].

However, there are comparatively few D-MAD techniques that exploit the many different deep learning techniques [77], [86]. Standard network architectures have been investigated, such as AlexNet. Auto-encoder convolutional networks have been investigated as a method to de-morph the reference image for comparison to a live individual [87]. Mutual information maximisation on disentangled representations can also be used as a D-MAD detector [88].

Generative Adversarial Networks (GANs) have been used for both D-MAD and S-MAD approaches. One instance for D-MAD de-morphs the reference image [89] in an attempt to provide a better facial match. Investigations into GANs for S-MAD have also recently been undertaken [90].

Due to the natural similarity/differential detection abilities of Siamese networks, it is a natural technique to apply to D-MAD. A couple of recent works have started to look into the use of Siamese Networks for D-MAD. Inception ResNet v1 [88] has been used as a Siamese network for D-MAD, as well as the recent SE-Resnet50 architecture being used as part of a double Siamese architecture [91] also for D-MAD. However, attention has not been paid to the usage of CNNs within a Siamese network in the context of D-MAD, nor an analysis to determine how well these Siamese networks compare to the same architecture in an S-MAD environment.

There are few datasets that can be used for D-MAD due to the requirement to have multiple images for the same individual. Two such datasets are FERET [39], [40] and Face Recognition Grand Challenge v2.0 [92] that both have content that matches this requirement.

4.2 Problem Definition

A mathematical definition and objective function of the problem can be found in Section 4.2.1 with a description of the dataset used in Section 4.2.2. The statistical equations used for evaluation can be found in Section 4.2.3.

4.2.1 Mathematical Definition

Let x_r be the reference image of the individual to be identified - this can either be a genuine image of the individual or a morphed image. x_l is a secondary image of the individual ideally from a live source. The objective function of a D-MAD system is to determine y - whether x_r is a morphed image or not (Eqtn. 4.1).

$$y(x_r, x_l) = \begin{cases} 1 & \text{if } x_r = \text{morphed image} \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

4.2.2 Datasets

In order to effectively test D-MAD techniques, a database that comprises of multiple images per individual is required for both x_r and x_l . The FERET database [39], [40] is one such suitable database for this purpose with 995 individuals, with multiple images each ¹.

The database has been separated into 3 separate sets: training (70% - 695), validation (10% - 100) and testing (20% - 200). The sets are generated in such a way that morphs from individuals cannot be found between the different sets (i.e. if individual 993 was in the training set, they cannot appear in the validation or testing set). Within the training set, each individual has been randomly matched up with 10 other individuals to create a total of 10 morphs per individual (an overall total of 3465 morphs). A morph is generated for every combination of individuals for the validation (9000 morphs) and testing (39800 morphs) sets.

Figure 4.1 gives examples of the generated morphs from this dataset. The field (FM) and triangle (TM) morphs have been generated using the face morphing pipeline described in [82] - both approaches use facial landmarks to align the face images, blend them together, and finally insert the segmented part of the blended imagery into one of the aligned input images. The triangle based morphing approach uses the feature selection and pre-processing as described in [83] that fuses and adds some points to the facial landmarks, to avoid flat and long triangles.

¹Portions of the research in this thesis use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office.

The average (AM) method has been generated using a modified open source face morpher [93] (modified to blend the images into the background) - this uses a similar technique to the TM, although produces different results.

Once generated, the images were automatically cropped to the region surrounding the head as required for passports and resized to 413x532 - then being compressed into JPEG2000 image format with an automatically determined compression ratio in order to fit into the 22kb requirement of the passport chip. The overall life cycle for morphed facial images within passports can be found [94].

4.2.3 Evaluation Metrics

The three metrics used in the comparisons are Attack Presentation Classification Error Rate (APCER) a.k.a. false positive rate (Equation 4.2), Bonafide Presentation Classification (BPCER) a.k.a. false negative rate (Equation 4.3) and weighted Average Classification Error Rate (Equation 4.4). In the equations: TP is true positives, FP is false positives, TN is true negatives and FN is false negatives. With all these metrics, the lower the score the better.

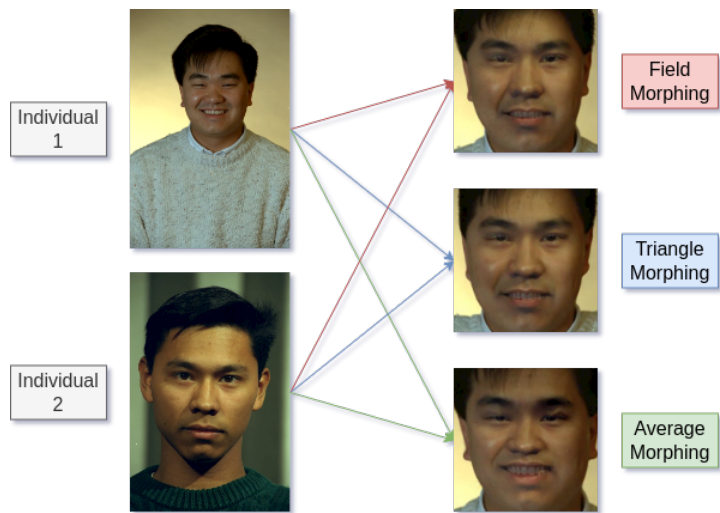
$$\text{APCER} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (4.2)$$

$$\text{BPCER} = \frac{\text{FN}}{\text{FN} + \text{TP}} \quad (4.3)$$

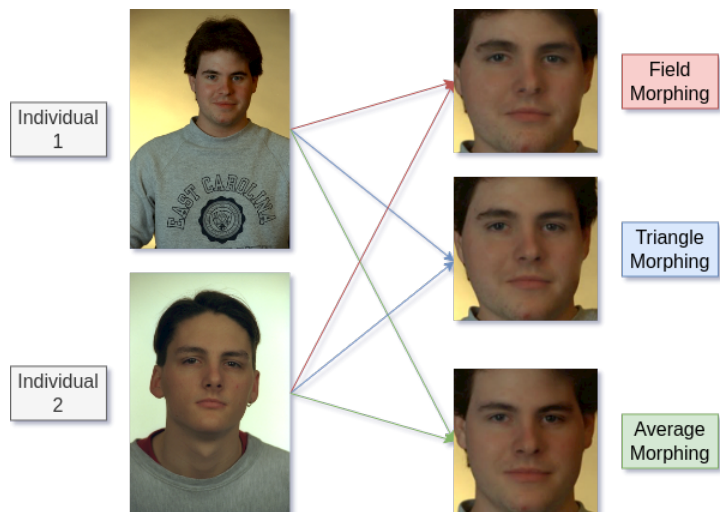
$$\text{ACER} = \frac{\text{APCER} + \text{BPCER}}{2} \quad (4.4)$$



(a) Morphing Example 1



(b) Morphing Example 2



(c) Morphing Example 3

Figure 4.1: Examples of the output of field, triangle, and average morphing techniques on three individuals from the FERET dataset [39], [40].

4.3 Siamese Model Description

The concept of a Siamese network [1] is to create a network of two parts. The first part is a feature network f that is used to extract out feature vectors F_r and F_l from inputs x_r and x_l - f is used with the same architecture and weights for each input. The difference of these feature vectors ($F_r - F_l$) are then calculated to be passed towards a final layer Y (or layers). In this case: the output layer Y provides a single output y that predicts whether the test is a morph attack or not (Eqtn. 4.1). This description is represented in Eqtn. 4.5. A pictorial representation of this, using a VGG19 architecture as an example, can be seen in Figure 4.2.

$$y(x_r, x_l) = Y(f(x_r) - f(x_l)) \quad (4.5)$$

The process to convert an existing network into a ‘feature’ network f for a Siamese network involves modifying the original architecture to extract out the relevant parts. This differs for each existing network - however, a general rule of thumb is to either remove the last layer and replace it with the remainder of the Siamese architecture, or to use the network in its entirety. In the cases below, only the layer defining the number of outputs of the network has been replaced.

Three networks defining different eras of Convolutional Neural Networks have been examined for this task: AlexNet (Figure 4.3a), VGG-19 (Figure 4.3b), and ResNet50 (Figure 4.3c).

A Siamese network can be guaranteed to train not only on the reference face x_r , but on both images due to the nature of its shared weights. Both the reference passport image x_r and the separate face image x_l will go through the same-weighted feature networks f before being differenced and passed to the final layer Y . This means that the final layer only sees one set of values once merged, so it can only look at the combination of features for its final task.

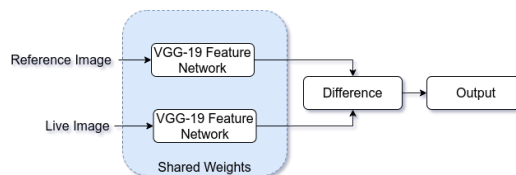
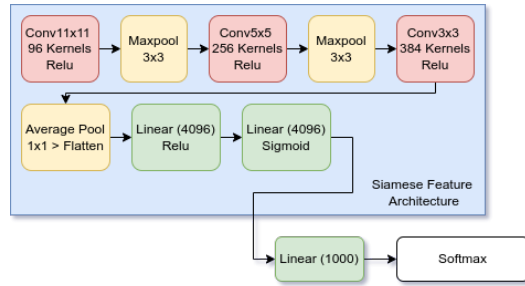
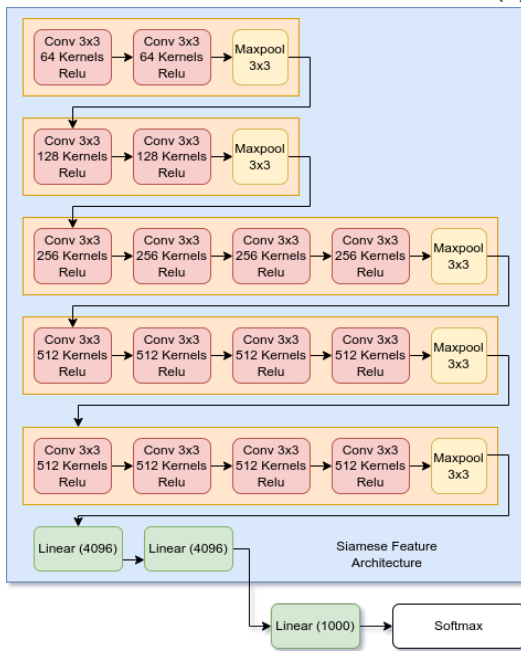


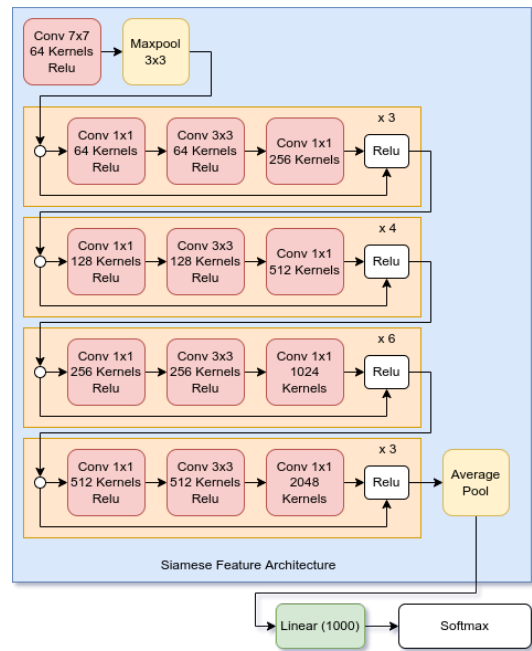
Figure 4.2: Siamese Differential Morphing Attack Detector Network Architecture with VGG-19 as the Feature Network



(a) AlexNet



(b) VGG-19



(c) ResNet50

Figure 4.3: Siamese Feature Architectures for Differential Morphing Attack Detection

4.4 Methodology

The overall input to the Siamese networks is two separate images: one is the reference image supplied from the passport x_r and one is a separate face image of the person being identified x_l . The images are cropped and resized as mentioned in Section 4.2.2 - then they are normalised to between 0 and 1. The overall training of the network foregoes testing for facial similarity i.e. it is not a face matcher. The training is purely to detect whether the provided reference is a morphed image or not.

It is necessary to consider the combination of x_r and x_l during the training process in order to ensure a balanced training between morphs and non-morphs. If not addressed, the networks may optimise into a local minima where it only outputs a single value (morph or non-morph depending on the imbalance), instead of correctly learning to differentiate between the two types of input. Each morphed image \mathbf{M} is a combination of two individuals (1 and 2), with each individual in the dataset having two images (a and b). As such, the exemplars are able to be presented as follows to keep a balanced training set: $\mathbf{M} \rightarrow 1a$, $\mathbf{M} \rightarrow 2a$, $1a \rightarrow 1b$, and $2a \rightarrow 2b$.

A Bayes search is performed on learning rate and weight decay for the Normalized Direction Preserving ADAM [16] optimiser. The input images are randomly flipped during training to help achieve robustness. The networks were trained for a total of 50 epochs with an MSE loss function. For each morphed reference image x_r , a total of 6 exemplars are passed through training - this includes comparison to each individual from the morph, as well as comparisons to each individual's secondary images. The validation set was used to determine a confidence threshold for the model, and the best epoch was chosen based on the ACER (Equation 4.4). The best model was then chosen from the Bayes search with this score.

Once the training is complete, the model is evaluated on the validation set to determine the threshold t where ACER is the lowest. If there are multiple thresholds where the ACER is 0, the final threshold t is selected at the midpoint of the largest range of 0s.

4.5 Results and Discussions

Table 4.1 shows the breakdown of results from all the detectors trained, as well as an equivalent S-MAD detector [82]. The metrics used are described in Section 4.2.3. The table shows the four models trained on either one of the three training datasets or the MIXED dataset containing all training datasets, with their results on each dataset to test their generalisability across datasets they have not seen. Bold values indicate the results on the same morphing technique that the

Method	Trained Set	Field Morphing			Triangle Morphing			Average Morphing		
		ACER	APCER	BPCER	ACER	APCER	BPCER	ACER	APCER	BPCER
S-MAD VGG-19	FM	0.6357 %	0.7714 %	0.5000 %	0.7651 %	1.0302 %	0.5000 %	1.4460 %	2.3920 %	0.5000 %
	TM	0.4322 %	0.3643 %	0.5000 %	0.4384 %	0.3769 %	0.5000 %	1.1809 %	1.8618 %	0.5000 %
	AM	3.5766 %	5.6533 %	1.5000 %	3.6922 %	5.8844 %	1.5000 %	0.9786 %	0.4573 %	1.5000 %
	MIXED	0.3756 %	0.2513 %	0.5000 %	0.3894 %	0.2789 %	0.5000 %	0.5276 %	0.5553 %	0.5000 %
D-MAD AlexNet	FM	4.3084 %	1.1168 %	7.5000 %	4.3524 %	1.2048 %	7.5000 %	7.2538 %	7.0075 %	7.5000 %
	TM	9.4441 %	4.3882 %	14.5000 %	9.5729 %	4.6457 %	14.5000 %	10.9089 %	7.3178 %	14.5000 %
	AM	3.8612 %	6.2224 %	1.5000 %	3.6790 %	5.8580 %	1.5000 %	1.6369 %	1.7739 %	1.5000 %
	MIXED	0.8907 %	1.2814 %	0.5000 %	0.8794 %	1.2588 %	0.5000 %	0.6049 %	0.7098 %	0.5000 %
D-MAD VGG-19	FM	0.0276 %	0.0553 %	0.0000 %	0.0188 %	0.0377 %	0.0000 %	0.0075 %	0.0151 %	0.0000 %
	TM	0.0044 %	0.0088 %	0.0000 %	0.0038 %	0.0075 %	0.0000 %	0.0006 %	0.0013 %	0.0000 %
	AM	0.4906 %	0.9812 %	0.0000 %	0.4987 %	0.9975 %	0.0000 %	0.0069 %	0.0138 %	0.0000 %
	MIXED	0.0038 %	0.0075 %	0.0000 %	0.0019 %	0.0038 %	0.0000 %	0.0000 %	0.0000 %	0.0000 %
D-MAD ResNet50	FM	0.0013 %	0.0025 %	0.0000 %	0.0013 %	0.0025 %	0.0000 %	0.0107 %	0.0214 %	0.0000 %
	TM	0.0000 %	0.0000 %	0.0000 %	0.0000 %	0.0000 %	0.0000 %	0.0038 %	0.0075 %	0.0000 %
	AM	0.0013 %	0.0025 %	0.0000 %	0.0019 %	0.0038 %	0.0000 %	0.0013 %	0.0025 %	0.0000 %
	MIXED	0.0000 %	0.0000 %	0.0000 %	0.0000 %	0.0000 %	0.0000 %	0.0006 %	0.0013 %	0.0000 %

Table 4.1: Results of Morphing Attack Detectors. All scores provided are error rates - lower values across all scores are better. Bold values indicate the results of the morphing type that the model was trained on.

model was trained on.

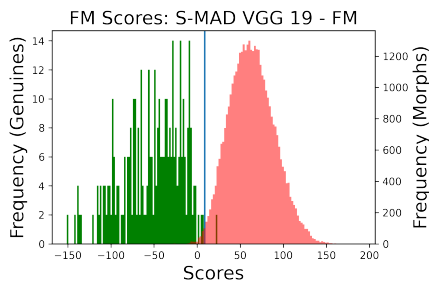
Regarding the models trained and tested on the same morphing technique, D-MAD AlexNet performs the worst with all its error rates being above 1% (ACERs: 4.3%, 9.6%, and 1.6%). All the other detectors were able to get less than a 1% error rate, with S-MAD VGG-19 (ACERs: 0.64%, 0.44%, and 0.98%), D-MAD VGG-19 (ACERs: 0.028%, 0.0038%, and 0.0069%), with the best being D-MAD ResNet50 (ACERs: 0.0013%, 0%, 0.0013%). These results show that a D-MAD Siamese network of the same architecture outperforms its S-MAD variant, as well as the higher performance of the newer designed CNNs/ResNets.

In terms of generalisability - all 4 architectures were capable of (for the most part) achieving similar error rates when testing on other datasets, albeit slightly worse due to having never seen any of the training data. In some cases, a model was able to outperform on a dataset it had not been trained on - for instance, D-MAD VGG-19's TM model was able to outperform on the FM (FM ACER 0.028% c.w. TM ACER 0.0044%) and AM (AM ACER 0.0069% c.w. TM ACER 0.0006%) datasets. While this pattern does appear also for the D-MAD ResNet50 with the TM model on the FM dataset, there are no wider patterns regarding models trained on the TM dataset.

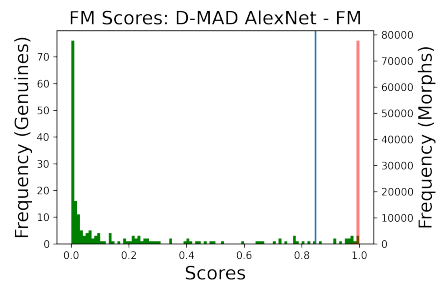
When trained across multiple datasets (MIXED), we can see that all architectures had an improvement with the most significant being the AlexNet model (ACERs: 4.3% \rightarrow 0.89%, 9.6% \rightarrow 0.88% and 1.6% \rightarrow 0.60%), bringing the error rates down to below 1% - to a similar level of error as the specifically trained S-MAD VGG-19s. For the S-MAD VGG-19 MIXED, it was able to nearly halve two out of three error rates while having a significant improvement on the third (ACERs: 0.64% \rightarrow 0.38%, 0.44% \rightarrow 0.39% and 0.98% \rightarrow 0.53%).

Figures 4.4, 4.5 and 4.6 show the score distributions of the FM, TM, and AM testing datasets respectively. Green indicates instances where the reference image x_r is genuine and red is a morph. The blue line indicates the chosen threshold t by determining the best ACER from the validation set. The results for the S-MAD models are in a different range as their final layers are trained to go through a Softmax; then the final score is determined by the value of the morphed output neuron minus the genuine output neuron.

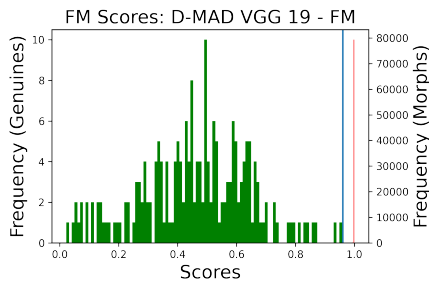
Both the D-MAD VGG-19 and D-MAD ResNet50 architectures are clearly capable of identifying the morphed reference images x_r , as the vast majority of scores provide a 1.0 with only comparatively few getting a lower score - this is clear from the early epochs of training. While the D-MAD AlexNet is less capable, it is still able to keep the morphed exemplars on the higher



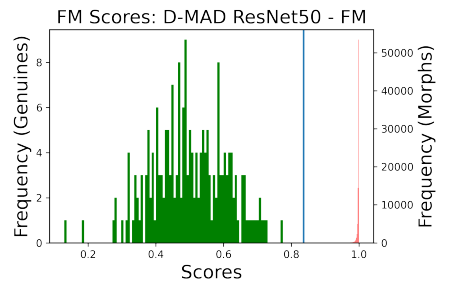
(a) S-MAD VGG-19



(b) D-MAD AlexNet

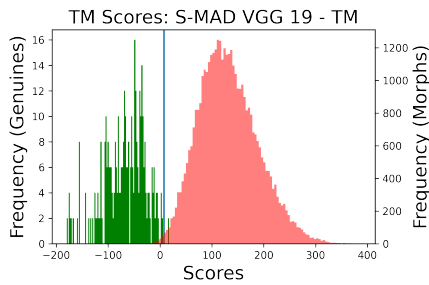


(c) D-MAD VGG-19

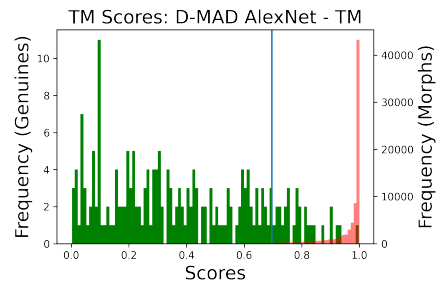


(d) D-MAD ResNet50

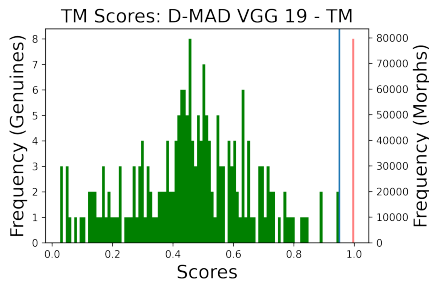
Figure 4.4: Score distributions for the Field Morphing (FM) dataset. Green represents a genuine label and red represents a morph label. The blue line is threshold t determined from validation set.



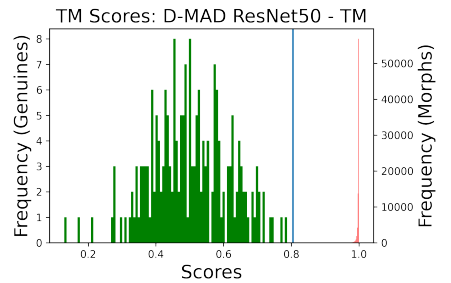
(a) S-MAD VGG-19



(b) D-MAD AlexNet



(c) D-MAD VGG-19



(d) D-MAD ResNet50

Figure 4.5: Score distributions for the Triangle Morphing (TM) dataset. Green represents a genuine label and red represents a morph label. The blue line is threshold t determined from validation set.

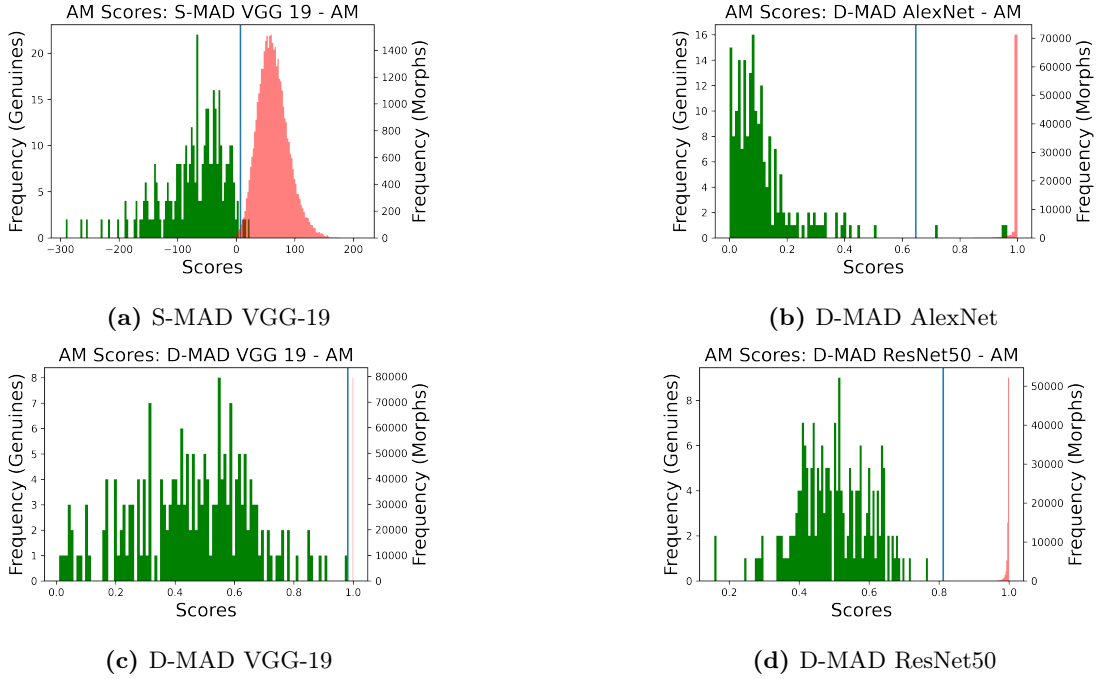


Figure 4.6: Score distributions for the Average Morphing (AM) dataset. Green represents a genuine label and red represents a morph label. The blue line is threshold t determined from validation set.

end of the score boundary. The primary struggles of the training process are actually bringing the score of the genuine exemplars down closer to the 0.0 mark. This is likely due to the number of genuine exemplar comparisons being far lower than the morph exemplar comparisons, despite efforts to mitigate by ensuring a more equal set of comparisons during training.

Table 4.2 shows the results from running the models trained on the TM dataset using the StirTrace [95] benchmarking. Three types of noise and six types of image modifications were generated for the entirety of the testing set. The error rates for the noisy imagery for three out of the four models show they are not robust at all to noise, with S-MAD VGG-19 (ACERs: 45.3%, 2.56%, 49.9%), D-MAD AlexNet (ACERs: 47.8%, 11.4%, 49.8%) and D-MAD VGG-19 (ACERs: 47.4%, 5.29%, and 46.7%) having extremely high error rates when compared to their original results. The VGG-19 S-MAD and D-MAD variants both show similar ACERs, however this comes across differently in the APCER and BPCERs - the networks have trained to place all scores at different ends of the possible score range. The D-MAD ResNet50 (ACERs: 4.16%, 0.0631%, 1.73%) was however able to keep a relatively low error rate in comparison to the others, although still showed a significant difference to unmodified images. Overall, Additive Noise 3 had a significantly lower effect on the results as compared to Additive Gaussian Noise 3 and Salt and Pepper 3, which put the models almost in line with randomly guessing - whereas on the ResNet50 model the noise has far less of an effect.

Noise / Modification Method	S-MAD VGG-19			D-MAD AlexNet			D-MAD VGG-19			D-MAD ResNet50		
	ACER	APCER	BPCER	ACER	APCER	BPCER	ACER	APCER	BPCER	ACER	APCER	BPCER
Additive Gaussian Noise 3	45.3%	90.5%	0.0%	47.8%	3.0%	92.7%	47.4%	2.17%	92.6%	4.16%	2.17%	6.14%
Additive Noise 3	2.56%	1.13%	4.0%	11.4%	5.5%	17.4%	5.29%	1.09%	9.49%	0.0631%	0.0%	0.126%
Salt and Pepper 3	49.9%	99.7%	0.0%	49.8%	0.5%	99.1%	46.7%	2.17%	91.3%	1.73%	1.09%	2.37%
Double Scale 50	2.66%	0.327%	5.0%	7.4%	5.5%	9.31%	0.0148%	0.0%	0.0297%	0.00223%	0.0%	0.00445%
Median Cut 3	18.3%	0.035%	36.5%	4.73%	3.0%	6.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Remove Columns 50	1.58%	0.656%	2.5%	6.83%	5.5%	8.15%	0.00445%	0.0%	0.00891%	0.0%	0.0%	0.0%
Remove Lines 50	1.58%	0.661%	2.5%	6.75%	5.5%	8.0%	0.00148%	0.0%	0.00297%	0.000742%	0.0%	0.00148%
Rotation -5°	8.77%	0.043%	17.5%	5.56%	5.5%	5.62%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
X Stretching 1.035	2.86%	0.226%	5.5%	6.53%	5.5%	7.56%	0.00297%	0.0%	0.00594%	0.000742%	0.0%	0.00148%

Table 4.2: Noise and Image Modification Testing of Models via StirTrace [95] on the Triangle Morphing Dataset

Both S-MAD VGG-19 and D-MAD AlexNet on the six image modifications are much less affected in their error rates than for noisy imagery, although there are some noted exceptions. S-MAD VGG-19 struggles with Median Cut 3 (ACER: 18.3%) and Rotation -5° (ACER 8.77%), with the rest being below 3%. The D-MAD AlexNet has higher error rates on four of the six modifications, however has a lower overall range (5.62% \rightarrow 9.31%) and doesn't struggle on the same types as the S-MAD VGG-19. Both the D-MAD VGG-19 and D-MAD ResNet50 were less susceptible to these types of modifications, with the D-MAD VGG-19's highest error rate being $\sim 0.03\%$ and the D-MAD ResNet50s highest error rate being $\sim 0.004\%$. This shows that the combination of image features (F_r and F_l) for VGG-19 is able to make it more robust to simple image modifications.

4.6 Deployment

The ResNet50 model was deployed in the EU Horizon 2020 D4FLY project [96]. The project's aims were to develop a set of tools and systems to address emerging threats in document and identity verification for border authorities seek to improve their capabilities and capacities. Two scenarios were addressed: land/sea/air border when passengers are leaving their transportation vehicle and pass through border controls on foot through a biometric corridor, as well as a coach scenario where travellers could be checked without exiting the vehicle. Both scenarios employed the use of an enrolment kiosk where the traveller registers with the system where the D-MAD system was deployed. Figure 4.7 shows both the enrolment kiosk (left) and the biometric corridor (right).

The purpose of the enrolment kiosk is to register travellers into its system for verification using novel biometrics. Prospective travellers use their passport to register with a number of biometrics being collected. A number of checks are performed during the enrolment process to both determine that the individual standing in front of the kiosk matches the information in the passport via facial recognition, as well as various checks to determine if the provided passport is



Figure 4.7: D4FLY Enrolment Kiosk (Left) and Biometric Corridor (Right)

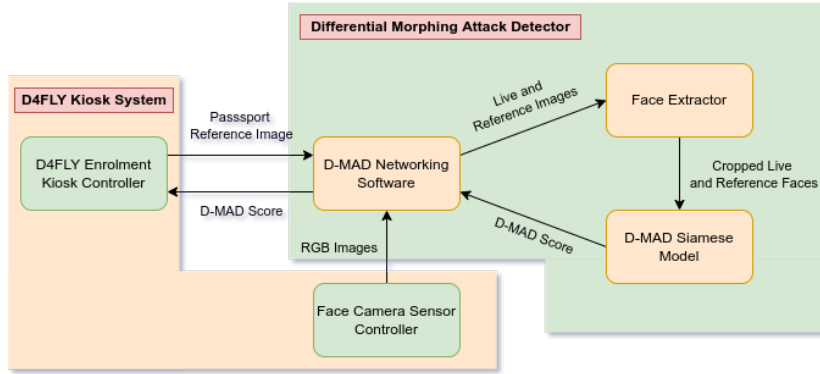


Figure 4.8: D-MAD Software Architecture for D4FLY

not a forgery in some form. S-MAD is a possibility for such a scenario as it can directly check the image in the passport - however, this is also a perfect opportunity to check the document against the live individual standing in front of the kiosk for D-MAD at the same time.

A protobuf interface was implemented to both communicate with the camera sensor software in the kiosk, and to receive commands with the required image from the passport. A piece of software in Rust was developed to handle this communication. The facial image received is the full image from the camera, so the correct facial region was extracted using a custom language interface to Visage’s FaceTrack SDK [97]. Once the individual is facing the sensor correctly, the image is cropped and passed through a Python interface to PyTorch for processing. The processed score and final decision is returned to the networking software which then forwards the score onto the Kiosk to either accept/reject the individual. A diagram of this process can be found in Figure 4.8.

4.7 Conclusions

This chapter proposes a set of novel Siamese networks for the purpose of D-MAD. It has been demonstrated that convolutional Siamese networks are capable of performing D-MAD, with adaptations of more recent models being better than earlier ones. The best model tested is ResNet50 (ACERs: 0% \rightarrow 0.001%) that is able to use its residual information model to create a better understanding of morphed imagery - and is a clear improvement on the VGG-19 model that also has good error rates (ACERs: 0.007% \rightarrow 0.02%). The ResNet50 model (\sim 23 million parameters) is a much smaller network than VGG-19 (\sim 143 million) and AlexNet (\sim 64 million), and can outperform both - proving that the residual information gives it a much better capability in this scenario. The D-MAD VGG-19 is capable of out-performing its S-MAD variant (ACERs: 0.4% \rightarrow 0.98%), demonstrating that having the additional information obtained from a secondary live image x_l of the individual is beneficial to determining whether the reference x_r image contains

a morphed image. The much smaller AlexNet model has much higher error rates (ACERs: 1.6% → 9.6%). All models were also able to improve when trained across multiple datasets with ACERs all below 1%, with both D-MAD ResNet50 and D-MAD VGG-19 near 0% error rate.

The trained models are however not robust to noisy imagery, and to a lesser extent various image modifications. All AlexNet and VGG-19 models are heavily affected by noise with up to nearly 50% ACERs; however the ResNet50 is much more capable at handling the noise (ACERs < 4.2%). All the D-MAD models are resistant to the other image modifications, keeping consistent ACERs with results based on unmodified images, with the S-MAD VGG-19 error rates being worse than completely unmodified imagery.

Chapter 5

Conclusions and Future Work

This thesis investigates the usage of Siamese networks in a few different forms across three land border security and surveillance scenarios: vehicle sub-type and make/model classification, trajectory pattern extraction and classification, and differential morphing attack detection. The following overall research question was posed:

- Are Siamese networks an appropriate and effective tool for the following security and surveillance tasks?
 - Vehicle Sub-Type and Make/Model Classification
 - Trajectory Pattern Extraction and Classification
 - Differential Morphing Attack Detection

The section will be split into discussions regarding these three topics - Section 5.1 for Vehicle Classification, Section 5.2 for Trajectory Pattern Extraction and Classification, and Section 5.3 for Differential Morphing Attack Detection. These are followed by a final conclusion in Section 5.4. The following questions were asked at the beginning for each topic:

- Is a Siamese network an appropriate tool for the scenario i.e. is it capable of performing robustly in the context of the defined task?
- Is a particular model able to outperform comparable/state-of-the-art methodologies as determined using appropriate benchmarking criteria?

5.1 Vehicle Sub-Type and Make/Model Classification

The first part of this thesis investigates the usage of Siamese networks for two different classification tasks: sub-type and make/model. Sub-type refers to the overall shape of the vehicle, whereas the make/model is the particular brand and model of the vehicle.

- **Is a Siamese network an appropriate tool for the scenario i.e. is it capable of performing robustly in the context of the defined task?**

A Siamese network is a type of network aimed to provide a similarity score between two exemplars. In order to achieve a classification task with a Siamese network, an input will need to be compared to a secondary image. This was thought to be a suitable application because of a large number of classes with an increasing number over time. In this vehicle scenario, the average class images of the training set were used for this secondary image.

For the datasets with a small number of classes, the proposed ResNet50 model was able to perform well with a sub-type f1-score of 0.933 and a Set 300 (number of classes with more than 300 exemplars) f1-score of 0.968. When the number of classes start to increase, the overall scores decrease to an overall 0.799 for the set, with the largest number of classes with the lowest score being 0.381.

There are multiple reasons as to why the datasets with a larger number of classes do not work as well. In order to circumvent the issue of over-training on classes with a large number of exemplars, a balanced training regime was used, only choosing a number of exemplars per epoch equivalent to the class with the smaller number of exemplars. This came as a detriment to a number of classes with no clear discernable pattern other than the number of classes to comparison.

In order to train the ResNet50 model to have good results, it was necessary to train the feature section of the Siamese network separately as an auto-encoder. Separating the training into two phases helped the model improve its performance significantly.

Two other models were examined based on AlexNet and VGG-19. Neither of these models were able to train well for any dataset despite the VGG-19 having reasonable looking output during the auto-encoder phase.

Siamese networks can work with a small number of classes when trained using an average class image as the secondary image - however, this depends on the model used. Only a residual

neural network in this case was able to perform well. The model's performance fell short when used on datasets with a larger number of classes.

- **Is a particular model able to outperform comparable/state-of-the-art methodologies as determined using appropriate benchmarking criteria?**

The proposed ResNet50 Siamese network was able to outperform the compared methods for the sub-type dataset, and was the second best performer on the make/model Set 300 (the set with the smallest number of classes). This model however falls far behind for sets with larger number of classes.

- **Future work**

It is clear that the major weaknesses of the proposed residual Siamese based classifier is with the datasets with a large number of classes with a large majority of them having a small number of exemplars.

Due to the large number of parameters for deep neural networks, a large number of exemplars is required to achieve high performance. Further data collection to increase the number of exemplars per class would be one way of addressing this issue. However, this is not necessarily possible as there are only so many vehicles on the road. Data augmentation - the process of making small modifications to exemplars in the datasets to increase exemplar count - is another solution worth investigating.

An investigation into using different types of secondary image for the similarity check - if a dataset can be created along the lines of make/model/year-of-production rather than an overall average, a one-shot training approach has the potential of working.

5.2 Trajectory Pattern Extraction and Classification

A novel model was designed for the purposes of trajectory pattern extraction and classification. The primary difference for this contribution compared to those detailed in Chapters 2 and 4 is the lack of convolutional layers - providing a test of the Siamese concept with just fully connected McCulloch Pitts neurons.

- **Is a Siamese network an appropriate tool for the scenario i.e. is it capable of performing robustly in the context of the defined task?**

The proposed Siamese models train well across all four datasets, achieving f1-scores of 1.0 (Traffic Junction), 0.94 (Parking Lot), 0.91 (Students 003) and 0.94 (Train Station). In certain cases, models are capable of achieving high scores when trained on one dataset and tested on another e.g. Parking Lot model (0.94) \rightarrow Traffic Junction (0.90) and Students 003 (0.91) \rightarrow Traffic Junction (0.85). When a model is trained across all datasets, it only provides comparable results on Parking Lot (0.94) with Traffic Junction being the second closest (0.93).

The disparity between testing on unseen data resides within the type of trajectories contained within the dataset. Both Traffic Junction and Parking Lot are vehicle scenarios with simpler smooth trajectories - this provides a simpler separation for classes with overall similar properties. When it comes to the pedestrian cases (Students 003 and Train Station) the primary sets of trajectories with the most exemplars from one is perpendicular to the second. As such, it is understandable why the vehicle scenarios performed better cross-dataset than the pedestrian scenarios.

- **Is a particular model able to outperform comparable/state-of-the-art methodologies as determined using appropriate benchmarking criteria?**

The Siamese models were tested against 5 other methodologies with the most recent being the Movelets methodology. The proposed Siamese model was capable of matching the Movelets for the Traffic Junction dataset (both with f1-scores of 1.0), and outperforms it on the Parking Lot (0.94 \rightarrow 0.68) and Train Station (0.94 \rightarrow 0.90) datasets.

- **Further work**

While the models were able to train and test well on their own datasets, the primary issues were from testing on unseen datasets with different types of trajectories. Further work should

investigate a method to determine which combinations of clusters should be the primary focus of training, with a specific sub-set of these difficult clusters, before the other clusters that are clearly separable. This could not only provide a better understanding for the model to provide better separation for similar classes, but if difficult clusters were gathered from multiple datasets, this could provide a far more robust model.

Another solution worth investigating is data augmentation - as a number of classes have a very small amount of exemplars, augmenting the data for these classes could produce a more balanced training set.

5.3 Differential Morphing Attack Detection

Three proposed Siamese models with the feature encoding based on pre-existing networks were examined for the purpose of Differential Morphing Attack Detection with a comparison to a current non-Siamese model on just the single image. When compared to the previous two use cases, this scenario is a more traditional case for Siamese networks, as the task to perform is a comparison of two images. A unique aspect of the comparison for this contribution is that there are not many tasks where a comparison between a differential comparison and single image evaluation is possible.

- **Is a Siamese network an appropriate tool for the scenario i.e. is it capable of performing robustly in the context of the defined task?**

Three Siamese models were tested across three different morphing methodologies: field morphing, triangle morphing, and the open source average morphing techniques. The error rates for the VGG-19 and ResNet50 models were all $< 0.1\%$ - even when evaluated on morphing techniques that they had not seen. The ResNet50 model has a significantly lower error than the VGG-19 proving the residual neural networks capabilities as better than non-residual counterparts. The AlexNet model achieved higher error rates, but still relatively low average error rates $< 7.5\%$.

The models, however, were significantly affected by Gaussian and Salt and Pepper noise added to the imagery - AlexNet and VGG-19 placing most exemplars in one of either 'morph' or 'not'. The ResNet50 model was also affected, however to far less a degree with error rates of $< 4.2\%$.

The ResNet50 model was deployed for the final demonstration of the D4FLY project [96] and capable of performing within the provided time window for the processing in its enrolment

kiosk.

- **Is a particular model able to outperform comparable/state-of-the-art methodologies as determined using appropriate benchmarking criteria?**

The Siamese models were compared against a S-MAD variant of the VGG-19 model. Both the differential VGG-19 and ResNet50 models were capable of out-performing the S-MAD variant. The S-MAD also suffered from noisy imagery - with the D-MAD ResNet50 being the least affected.

- **Further work**

Improving the robustness of the current models is important for any real world deployment. As the current models are only trained with simple image augmentation techniques (flipping imagery), they have not had any chance to learn from these types of noise; adding noise into the training cycle should improve the overall scores and therefore make the models more robust. The primary candidate from the models tested would be the ResNet50 due to its already low error rates on robustness checks.

Testing other models is always an option to determine if they can perform better - an interesting study would be on other residual neural networks to determine if smaller models achieve a similar set of results, which would be beneficial to transaction times for a live scenario or mass processing.

5.4 Overall Conclusions and Future Work

This thesis has presented three different proposed solutions using Siamese networks for land border surveillance and security scenarios. Each scenario has also presented a different type of task for Siamese networks. Vehicle sub-type and make/model classification provided a scenario for image instance identification/classification. Trajectory pattern extraction and classification provided another scenario for classification; however, it was in the separate context of trajectories which required a different type of model to the convolutional networks used in the other application domains. Differential morphing attack detection returned to using imagery, but in the context of performing a strict process of comparing two images.

All three scenarios had tasks that worked well and tasks that did not. The proposed ResNet50 model worked the best for both the vehicle chapter and the D-MAD chapter - showing that

the advancements in neural network architecture design work within the context of Siamese networks. The proposed trajectory model outperformed its counterparts on three out of four datasets - showing that the concept not only works for convolutional networks, but also for networks purely containing McCulloch Pitts neurons.

The issues from the vehicle and trajectory chapter, while different, had a similar theme and future work despite their different types of input data. The vehicle chapter had poor results on datasets with a larger number of classes. The trajectory chapter had good results on individual datasets, but fell through on both cross-dataset training and in most cases unseen datasets. Future work for both would be investigating how to train Siamese networks on both similar and different looking classes/clusters. Data augmentation is another methodology that has the potential for improving the results for both chapters.

The proposed models from the face morphing chapter had significant issues with noise - excluding the ResNet50 model which only had minor issues. Improving on the data augmentation, with regards to noise added to the training datasets, would likely improve the overall robustness of the model.

Bibliography

- [1] J. Bromley, J. Bentz, L. Bottou, I. Guyon, Y. Lecun, C. Moore, E. Sackinger and R. Shah, “Signature Verification using a "Siamese" Time Delay Neural Network”, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, p. 25, 1st Aug. 1993. DOI: 10.1142/S0218001493000339.
- [2] R. K. McConnell, “Method of and apparatus for pattern recognition”, U.S. Patent 4567610A, 28th Jan. 1986. [Online]. Available: <https://patents.google.com/patent/US4567610/en> (visited on 23/06/2022).
- [3] M. Khalil-Hani and L. S. Sung, “A convolutional neural network approach for face verification”, in *2014 International Conference on High Performance Computing & Simulation (HPCS)*, Jul. 2014, pp. 707–714. DOI: 10.1109/HPCSim.2014.6903759.
- [4] L. Parzianello and A. Czajka, “Saliency-Guided Textured Contact Lens-Aware Iris Recognition”, in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Jan. 2022, pp. 330–337. DOI: 10.1109/WACVW54805.2022.00039.
- [5] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943, ISSN: 0007-4985. DOI: 10.1007/BF02478259.
- [6] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’12, Red Hook, NY, USA: Curran Associates Inc., 3rd Dec. 2012, pp. 1097–1105.
- [7] J. Schmidhuber, “Deep Learning in neural networks: An overview”, *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.
- [8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge”, *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 1st Dec. 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y.
- [9] K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition*, 10th Dec. 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385 [cs].

- [10] H. Robbins and S. Monro, “A Stochastic Approximation Method”, *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951, ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729586.
- [11] D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning representations by back-propagating errors”, *Nature*, vol. 323, no. 6088, pp. 533–536, 6088 Oct. 1986, ISSN: 1476-4687. DOI: 10.1038/323533a0.
- [12] J. Duchi, E. Hazan and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”, *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011, ISSN: 1533-7928. [Online]. Available: <http://jmlr.org/papers/v12/duchi11a.html> (visited on 25/06/2022).
- [13] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method”, *CoRR*, vol. abs/1212.5, 2012, ISSN: 1212.5701. [Online]. Available: <http://arxiv.org/abs/1212.5701>.
- [14] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, 29th Jan. 2017, [Online]. Available: <http://arxiv.org/abs/1412.6980> (visited on 09/04/2020).
- [15] G. E. Hinton, “Overview of mini-batch gradient descent”, 2012, [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (visited on 01/11/2022).
- [16] Z. Zhang, L. Ma, Z. Li and C. Wu, “Normalized Direction-preserving Adam”, 17th Sep. 2018.
- [17] Y. Bengio, “Learning Deep Architectures for AI”, *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009, ISSN: 2200000006. DOI: 10.1561/2200000006. [Online]. Available: <http://www.nowpublishers.com/article/Details/MAL-006>.
- [18] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction”, in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, Aug. 2004, 28–31 Vol.2. DOI: 10.1109/ICPR.2004.1333992.
- [19] H. Griffiths, “Number of cars being cloned quadruples in six years”, *Auto Express*, 4th Mar. 2019. [Online]. Available: <https://www.autoexpress.co.uk/car-news/106500/number-of-cars-being-cloned-quadruples-in-six-years> (visited on 19/06/2022).
- [20] D. K. German, “Identity crisis: 90,000 vehicles with false plates could be on UK roads”, *The Telegraph*, 20th Dec. 2018, ISSN: 0307-1235. [Online]. Available: <https://www.telegraph.co.uk/cars/comment/identity-crisis-90000-vehicles-false-plates-could-uk-roads/> (visited on 19/06/2022).

- [21] J. Boyle and J. Ferryman, “Vehicle subtype, make and model classification from side profile video”, in *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2015, pp. 1–6. DOI: 10.1109/AVSS.2015.7301783.
- [22] X. Wang, K. Tieu and E. L. Grimson, “Correspondence-Free Activity Analysis and Scene Modeling in Multiple Camera Views”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 56–71, Jan. 2010, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.241.
- [23] B. Zhou, X. Wang and X. Tang, “Understanding collective crowd behaviors: Learning a Mixture model of Dynamic pedestrian-Agents”, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 2871–2878. DOI: 10.1109/CVPR.2012.6248013.
- [24] J. Peng, F. Qiu, J. See, Q. Guo, S. Huang, L.-Y. Duan and W. Lin, “Tracklet Siamese Network with Constrained Clustering for Multiple Object Tracking”, in *2018 IEEE Visual Communications and Image Processing (VCIP)*, Dec. 2018, pp. 1–4. DOI: 10.1109/VCIP.2018.8698623.
- [25] L. Patino and J. Ferryman, “Multiresolution semantic activity characterisation and abnormality discovery in videos”, *Applied Soft Computing*, vol. 25, pp. 485–495, 1st Dec. 2014, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2014.08.039.
- [26] M. Andersson, R. Johansson, K.-G. Stenborg, R. Forsgren, T. Cane, G. Taberski, L. Patino and J. Ferryman, “The IPATCH System for Maritime Surveillance and Piracy Threat Classification”, in *2016 European Intelligence and Security Informatics Conference (EISIC)*, Aug. 2016, pp. 200–200. DOI: 10.1109/EISIC.2016.054.
- [27] S. Dai, L. Li and Z. Li, “Modeling Vehicle Interactions via Modified LSTM Models for Trajectory Prediction”, *IEEE Access*, vol. 7, pp. 38 287–38 296, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2907000.
- [28] I. Choi, H. Song and J. Yoo, “Deep Learning Based Pedestrian Trajectory Prediction Considering Location Relationship between Pedestrians”, in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, Feb. 2019, pp. 449–451. DOI: 10.1109/ICAIIIC.2019.8669009.
- [29] H. Xue, D. Q. Huynh and M. Reynolds, “SS-LSTM: A Hierarchical LSTM Model for Pedestrian Trajectory Prediction”, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2018, pp. 1186–1194. DOI: 10.1109/WACV.2018.00135.

- [30] J. Boyle, T. Nawaz and J. Ferryman, “Deep trajectory representation-based clustering for motion pattern extraction in videos”, in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078509.
- [31] M. Lewis and P. Statham, “CESG biometric security capabilities programme: Method, results and research challenges”, in *Proceedings Biometrics Consortium Conference (BCC)*, 2004. DOI: 10.1007/978-981-15-8342-1_6.
- [32] “ISO19792 - Security evaluation of biometrics”, Standard, Mar. 2009.
- [33] M. Ferrara, A. Franco and D. Maltoni, “The magic passport”, in *IEEE International Joint Conference on Biometrics*, Sep. 2014, pp. 1–7. DOI: 10.1109/BTAS.2014.6996240.
- [34] R. Raghavendra, K. B. Raja and C. Busch, “Detecting morphed face images”, in *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, Sep. 2016, pp. 1–7. DOI: 10.1109/BTAS.2016.7791169.
- [35] Z. Bukovčiková, D. Sopiak, M. Oravec and J. Pavlovičová, “Face verification using convolutional neural networks with Siamese architecture”, in *2017 International Symposium ELMAR*, Sep. 2017, pp. 205–208. DOI: 10.23919/ELMAR.2017.8124469.
- [36] H. Wu, Z. Xu, J. Zhang, W. Yan and X. Ma, “Face recognition based on convolution siamese networks”, in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Oct. 2017, pp. 1–5. DOI: 10.1109/CISP-BMEI.2017.8302003.
- [37] M. Evans, J. Boyle and J. Ferryman, “Vehicle Classification using Evolutionary Forests”, in *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, vol. 2, 1st Jan. 2012, pp. 387–393.
- [38] J. Boyle, T. Nawaz and J. Ferryman, “Using Deep Siamese networks for trajectory analysis to extract motion patterns in videos”, *Electronics Letters*, vol. 58, no. 9, pp. 356–359, 2022, ISSN: 1350-911X. DOI: 10.1049/e112.12460.
- [39] P. J. Phillips, H. Wechsler, J. Huang and P. J. Rauss, “The FERET database and evaluation procedure for face-recognition algorithms”, *Image and Vision Computing*, vol. 16, no. 5, pp. 295–306, 27th Apr. 1998, ISSN: 0262-8856. DOI: 10.1016/S0262-8856(97)00070-X.
- [40] P. Phillips, H. Moon, S. Rizvi and P. Rauss, “The FERET evaluation methodology for face-recognition algorithms”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000, ISSN: 1939-3539. DOI: 10.1109/34.879790.

- [41] B. C. Putra, B. Setiyono, D. R. Sulistyaningrum, Soetrisno and I. Mukhlash, “Moving Vehicle Classification Using Pixel Quantity Based on Gaussian Mixture Models”, in *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, Apr. 2018, pp. 254–257. DOI: 10.1109/CCOMS.2018.8463218.
- [42] S. Shi, Z. Qin and J. Xu, “Robust Algorithm of Vehicle Classification”, in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 3, Jul. 2007, pp. 269–272. DOI: 10.1109/SNPD.2007.79.
- [43] N. S. Thakoor and B. Bhanu, “Structural signatures for passenger vehicle classification in video”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1796–1805, 2013, ISSN: 1524-9050. DOI: 10.1109/TITS.2013.2269137.
- [44] Z. Ding and W. Mo, “Vehicle recognition by SDCD lamp sectional distance”, in *2021 7th International Conference on Computer and Communications (ICCC)*, Dec. 2021, pp. 1853–1858. DOI: 10.1109/ICCC54389.2021.9674475.
- [45] P. Dalka and A. Czyzewski, “Vehicle Classification Based on Soft Computing Algorithms”, in *Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing*, 2010, pp. 70–79. DOI: 10.1007/978-3-642-13529-3_9.
- [46] N. C. Mithun, N. U. Rashid and S. M. M. Rahman, “Detection and Classification of Vehicles From Video Using Multiple Time-Spatial Images”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1215–1225, 2012. DOI: 10.1109/TITS.2012.2186128.
- [47] S. Anuja Prasad and L. Mary, “A Comparative Study of Different Features for Vehicle Classification”, in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, Feb. 2019, pp. 1–5. DOI: 10.1109/ICCIDS.2019.8862136.
- [48] B. Abdillah, G. Jati and W. Jatmiko, “Improvement CNN Performance by Edge Detection Preprocessing for Vehicle Classification Problem”, in *2018 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Dec. 2018, pp. 1–7. DOI: 10.1109/MHS.2018.8887015.
- [49] E. U. Armin, A. Bejo and R. Hidayat, “Vehicle Type Classification in Surveillance Image based on Deep Learning Method”, in *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*, Nov. 2020, pp. 400–404. DOI: 10.1109/ICOIACT50329.2020.9332047.
- [50] L. Yang, P. Luo, C. C. Loy and X. Tang, “A large-scale car dataset for fine-grained categorization and verification”, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 3973–3981. DOI: 10.1109/CVPR.2015.7299023.

- [51] (5th Oct. 2011). Same car - different badges, [Online]. Available: <https://www.telegraph.co.uk/motoring/picturegalleries/8806776/Same-car-different-badge.html> (visited on 30/06/2022).
- [52] B. Zhang, “Reliable classification of vehicle types based on cascade classifier ensembles”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 322–332, Mar. 2013, ISSN: 1524-9050 VO - PP. DOI: 10.1109/TITS.2012.2213814.
- [53] M. A. Manzoor and Y. Morgan, “Vehicle Make and Model classification system using bag of SIFT features”, in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2017, pp. 1–5. DOI: 10.1109/CCWC.2017.7868475.
- [54] A. A. Jamil, F. Hussain, M. H. Yousaf, A. M. Butt and S. A. Velastin, “Vehicle Make and Model Recognition Using Bag of Expressions”, *Sensors (Basel, Switzerland)*, vol. 20, no. 4, p. 1033, 14th Feb. 2020, ISSN: 1424-8220. DOI: 10.3390/s20041033. pmid: 32075119.
- [55] P. Ajitha, J. S, Y. N. Krishna K and A. Sivasangari, “Vehicle Model Classification Using Deep Learning”, in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, Jun. 2021, pp. 1544–1548. DOI: 10.1109/ICOEI51242.2021.9452842.
- [56] I. O. de Oliveira, K. V. O. Fonseca and R. Minetto, “A Two-Stream Siamese Neural Network for Vehicle Re-Identification by Using Non-Overlapping Cameras”, in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 669–673. DOI: 10.1109/ICIP.2019.8803810.
- [57] Efficient Integrated Security Checkpoints, [Online]. Available: <https://cordis.europa.eu/project/id/217991> (visited on 30/06/2022).
- [58] L. Breiman, *Random Forests*. 2001, 5 pp.
- [59] R. Rifkin and A. Klautau, “In Defense of One-Vs-All Classification”, *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 1st Dec. 2004.
- [60] D. Makris and T. Ellis, “Learning semantic scene models from observing activity in visual surveillance”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 397–408, Jun. 2005, ISSN: 1083-4419. DOI: 10.1109/TSMCB.2005.846652.
- [61] I. Saleemi, L. Hartung and M. Shah, “Scene understanding by statistical modeling of motion patterns”, in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 2069–2076. DOI: 10.1109/CVPR.2010.5539884.

- [62] S. Calderara, A. Prati and R. Cucchiara, “Mixtures of von Mises Distributions for People Trajectory Shape Analysis”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 4, pp. 457–471, Apr. 2011, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2011.2125550.
- [63] T. Shu, S. Todorovic and S.-C. Zhu, “CERN: Confidence-Energy Recurrent Network for Group Activity Recognition”, 10th Apr. 2017, [Online]. Available: <http://arxiv.org/abs/1704.03058> (visited on 12/12/2018).
- [64] L. Patino, T. Nawaz, T. Cane and J. Ferryman, “PETS 2017: Dataset and Challenge”, in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul. 2017, pp. 2126–2132. DOI: 10.1109/CVPRW.2017.264.
- [65] W. Hu, X. Li, G. Tian, S. Maybank and Z. Zhang, “An Incremental DPMM-Based Method for Trajectory Clustering, Modeling, and Retrieval”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1051–1065, May 2013, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.188.
- [66] T. Nawaz, A. Cavallaro and B. Rinner, “Trajectory clustering for motion pattern extraction in aerial videos”, in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 1016–1020. DOI: 10.1109/ICIP.2014.7025203.
- [67] C. A. Ferrero, L. M. Petry, L. O. Alvares, C. L. da Silva, W. Zalewski and V. Bogorny, “MasterMovelets: Discovering heterogeneous movelets for multiple aspect trajectory classification”, *Data Mining and Knowledge Discovery*, vol. 34, no. 3, pp. 652–680, 1st May 2020, ISSN: 1573-756X. DOI: 10.1007/s10618-020-00676-x.
- [68] H. Ailin, L. Zhong and Z. Dechao, “Movement Pattern Extraction Based on a Non-parameter Sub-trajectory Clustering Algorithm”, in *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*, Mar. 2019, pp. 5–9. DOI: 10.1109/ICBDA.2019.8713239.
- [69] D. Yao, C. Zhang, Z. Zhu, J. Huang and J. Bi, “Trajectory Clustering via Deep Representation Learning”, *Neural Networks (IJCNN), International Joint Conference on*, pp. 3880–3887, 2017, ISSN: 9781509061822. DOI: 10.1109/IJCNN.2017.7966345.
- [70] J. Šochman and D. C. Hogg, “Who knows who - Inverting the Social Force Model for finding groups”, in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov. 2011, pp. 830–837. DOI: 10.1109/ICCVW.2011.6130338.
- [71] J. Bian, D. Tian, Y. Tang and D. Tao, “Trajectory Data Classification: A Review”, *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 4, pp. 1–34, 29th Aug. 2019, ISSN: 2157-6904, 2157-6912. DOI: 10.1145/3330138.

- [72] S. LaValle, M. Branicky and S. Lindemann, “On the Relationship between Classical Grid Search and Probabilistic Roadmaps”, *I. J. Robotic Res.*, vol. 23, pp. 673–692, 1st Aug. 2004, ISSN: 978-3-642-07341-0. DOI: 10.1007/978-3-540-45058-0_5.
- [73] L. Biewald, *Experiment Tracking with Weights and Biases*, Weights and Biases, 2020. [Online]. Available: <https://www.wandb.com/>.
- [74] G. Koch, R. Zemel and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition”, in *ICML Deep Learning Workshop*, vol. 2, Lille, 2015.
- [75] N. Anjum and A. Cavallaro, “Multifeature object trajectory clustering for video analysis”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1555–1564, 2008, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2008.2005603.
- [76] U. Scherhag, C. Rathgeb and C. Busch, “Towards Detection of Morphed Face Images in Electronic Travel Documents”, in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Apr. 2018, pp. 187–192. DOI: 10.1109/DAS.2018.11.
- [77] S. Venkatesh, R. Ramachandra, K. Raja and C. Busch, “Face Morphing Attack Generation and Detection: A Comprehensive Survey”, *IEEE Transactions on Technology and Society*, vol. 2, no. 3, pp. 128–145, Sep. 2021, ISSN: 2637-6415. DOI: 10.1109/TTS.2021.3066254.
- [78] U. Scherhag, C. Rathgeb and C. Busch, “Morph Detection from Single Face Image: A Multi-Algorithm Fusion Approach”, in *Proceedings of the 2018 2nd International Conference on Biometric Engineering and Applications*, ser. ICBEA '18, New York, NY, USA: Association for Computing Machinery, 16th May 2018, pp. 6–12, ISBN: 978-1-4503-6394-5. DOI: 10.1145/3230820.3230822.
- [79] U. Scherhag, L. Debiase, C. Rathgeb, C. Busch and A. Uhl, “Detection of Face Morphing Attacks Based on PRNU Analysis”, *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 1, no. 4, pp. 302–317, Oct. 2019, ISSN: 2637-6407. DOI: 10.1109/TBIOM.2019.2942395.
- [80] R. Ramachandra, S. Venkatesh, K. Raja and C. Busch, “Towards making Morphing Attack Detection robust using hybrid Scale-Space Colour Texture Features”, in *2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA)*, Jan. 2019, pp. 1–8. DOI: 10.1109/ISBA.2019.8778488.
- [81] R. Raghavendra, K. B. Raja, S. Venkatesh and C. Busch, “Transferable Deep-CNN Features for Detecting Digital and Print-Scanned Morphed Face Images”, in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul. 2017, pp. 1822–1830. DOI: 10.1109/CVPRW.2017.228.

- [82] C. Seibold, W. Samek, A. Hilsmann and P. Eisert, *Detection of Face Morphing Attacks by Deep Learning*. 26th Jul. 2017, p. 120, 107 pp., ISBN: 978-3-319-64184-3. DOI: 10.1007/978-3-319-64185-0_9.
- [83] —, “Accurate and robust neural networks for face morphing attack detection”, *Journal of Information Security and Applications*, vol. 53, p. 102 526, 1st Aug. 2020, ISSN: 2214-2126. DOI: 10.1016/j.jisa.2020.102526.
- [84] C. Seibold, A. Hilsmann and P. Eisert, “Focused LRP: Explainable AI for Face Morphing Attack Detection”, in *2021 IEEE Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Jan. 2021, pp. 88–96. DOI: 10.1109/WACVW52041.2021.00014.
- [85] P. Aghdaie, B. Chaudhary, S. Soleymani, J. Dawson and N. M. Nasrabadi, “Detection of Morphed Face Images Using Discriminative Wavelet Sub-bands”, in *2021 IEEE International Workshop on Biometrics and Forensics (IWBF)*, May 2021, pp. 1–6. DOI: 10.1109/IWBF50991.2021.9465074.
- [86] A. Makrushin and A. Wolf, “An Overview of Recent Advances in Assessing and Mitigating the Face Morphing Attack”, in *2018 26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 1017–1021. DOI: 10.23919/EUSIPCO.2018.8553599.
- [87] D. Ortega-Delcampo, C. Conde, D. Palacios-Alonso and E. Cabello, “Border Control Morphing Attack Detection With a Convolutional Neural Network De-Morphing Approach”, *IEEE Access*, vol. 8, pp. 92 301–92 313, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2994112.
- [88] S. Soleymani, A. Dabouei, F. Taherkhani, J. Dawson and N. M. Nasrabadi, “Mutual Information Maximization on Disentangled Representations for Differential Morph Detection”, in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2021, pp. 1730–1740. DOI: 10.1109/WACV48630.2021.00177.
- [89] F. Peng, L.-B. Zhang and M. Long, “FD-GAN: Face De-Morphing Generative Adversarial Network for Restoring Accomplice’s Facial Image”, *IEEE Access*, vol. 7, pp. 75 122–75 131, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2920713.
- [90] Z. Blasingame and C. Liu, “Leveraging Adversarial Learning for the Detection of Morphing Attacks”, in *2021 IEEE International Joint Conference on Biometrics (IJCB)*, Aug. 2021, pp. 1–8. DOI: 10.1109/IJCB52358.2021.9484383.
- [91] G. Borghi, E. Pancisi, M. Ferrara and D. Maltoni, “A Double Siamese Framework for Differential Morphing Attack Detection”, *Sensors*, vol. 21, p. 3466, 16th May 2021. DOI: 10.3390/s21103466.

- [92] P. Phillips, P. Flynn, T. Scruggs, K. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min and W. Worek, “Overview of the face recognition grand challenge”, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, Jun. 2005, 947–954 vol. 1. DOI: 10.1109/CVPR.2005.268.
- [93] A. Quek, *Face Morpher*, 5th Jan. 2019. [Online]. Available: https://github.com/alyssaq/face_morpher (visited on 02/08/2021).
- [94] C. Kraetzer, A. Makrushin, T. Neubert, M. Hildebrandt and J. Dittmann, “Modeling Attacks on Photo-ID Documents and Applying Media Forensics for the Detection of Facial Morphing”, in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, ser. IH & MMSec ’17, New York, NY, USA: Association for Computing Machinery, 20th Jun. 2017, pp. 21–32, ISBN: 978-1-4503-5061-7. DOI: 10.1145/3082031.3083244.
- [95] M. Hildebrandt and J. Dittmann, “From StirMark to StirTrace: Benchmarking pattern recognition based printed fingerprint detection”, in *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security*, ser. IH & MMSec ’14, New York, NY, USA: Association for Computing Machinery, 11th Jun. 2014, pp. 71–76, ISBN: 978-1-4503-2647-6. DOI: 10.1145/2600918.2600926.
- [96] Detecting Document frauD and iDentity on the fly, [Online]. Available: <https://cordis.europa.eu/project/id/833704> (visited on 30/06/2022).
- [97] *Visage FaceTrack*, Visage Technologies. [Online]. Available: <https://visagetechologies.com/facetrack/> (visited on 29/06/2022).